UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

MICROSOFT CORPORATION,
Petitioner,

v.

QOMPLX LLC,
Patent Owner.

IPR2026-00298
Patent: USP 12,143,424
Issued: November 12, 2024
Application No. 18/779,029
Filed: July 21, 2024

Title: RAPID PREDICTIVE ANALYSIS OF VERY LARGE DATA SETS USING THE DISTRIBUTED COMPUTATIONAL GRAPH

DECLARATION OF MARK CROVELLA

MICROSOFT CORP.
EXHIBIT 1003

TABLE OF CONTENTS

Page

I. INTRODUCTION AND ENGAGEMENT 5

II. BACKGROUND AND QUALIFICATIONS 5

III. MATERIALS CONSIDERED AND INFORMATION RELIED UPON REGARDING '424 PATENT 8

IV. UNDERSTANDING OF PATENT LAW 10

V. SUMMARY OF OPINIONS 13

VI. LEVEL OF SKILL IN THE ART, AND STATE OF THE ART 14

A. Person Of Ordinary Skill In The Art 14

B. State Of The Art 16

1. Distributed, Highly Scalable, Stream Processing Engines Were Commonly Used For Real-Time Data Processing 16

2. Stream Processing Engines Typically Used Directed Graphs To Organize And Manage Processing Data Streams 18

VII. THE '424 PATENT 20

A. The Claims' Earliest Effective Filing Date 20

B. The '424 Patent's Specification 20

VIII. CLAIM CONSTRUCTION 21

IX. GROUND 1: CLAIMS 1-3, 5, 7, 9-11, AND 13 ARE OBVIOUS OVER BARSNESS 22

A. Barsness (EX1004) 22

A. Claim 1 24

1. Element [1.1] 24

2. Element [1.2] 26

a) Element [1.2.a] 31

b) Element [1.2.b] 34
c) Element [1.2.c].....41
3. Element [1.3].....45
a) Element [1.3.a].....46
b) Element [1.3.b] 48
c) Element [1.3.c].....54
d) Element [1.3.d] 61
4. Element [1.4].....62
a) Element [1.4.a].....63
b) Element [1.4.b] 64
5. Element [1.5].....66
6. Element [1.6].....67
7. Element [1.7].....70
B. Claim 2 78
C. Claim 3 80
D. Claim 5 88
E. Claim 7 90
F. Claim 9 94
1. Element [9.1].....94
2. Element [9.2].....94
3. Element [9.3].....95
4. Element [9.4].....98
5. Element [9.5].....99
6. Element [9.6].....99

7. Element [9.7] 99

8. Element [9.8] 99

G. Claim 10 100

H. Claim 11 100

I. Claim 13 100

X. GROUND 2: CLAIMS 1-3, 5, 7, 9-11, AND 13 ARE OBVIOUS OVER BARSNESS IN VIEW OF CHAKRADHAR 101

A. Chakradhar (EX1005) 101

1. Chakradhar Is Supported By The Chakradhar Provisional 102

a) The Chakradhar Provisional Supports Claim 1 Of Chakradhar 102

b) The Chakradhar Provisional Supports Relied-Upon Subject Matter Of Chakradhar 104

B. Combining Barsness And Chakradhar 104

C. Claims 1-3, 5, 7, 9-11, And 13 106

XI. GROUND 3: CLAIMS 1-3, 5, 7, 9-11, AND 13 ARE OBVIOUS OVER BARSNESS IN VIEW OF SIRIPURAPU 111

A. Siripurapu (EX1007) 111

B. Combining Barsness And Siripurapu 114

C. Claims 1-3, 5, 7, 9-11, And 13 116

XII. GROUND 4: CLAIMS 1-3, 5, 7, 9-11, AND 13 ARE OBVIOUS OVER BARSNESS IN VIEW OF CHAKRADHAR AND SIRIPURAPU 117

I, Mark Crovella, do hereby declare as follows:

I. INTRODUCTION AND ENGAGEMENT

1. I have been retained as an independent expert on behalf of Microsoft Corporation in connection with the above-captioned Petition for *Inter Partes* Review ("IPR") to provide my analyses and opinions on certain technical issues related to U.S. Patent No. 12,143,424 (hereinafter "the '424 patent").

2. I am being compensated at my usual and customary rate for the time I spent in connection with this IPR. My compensation is not affected by the outcome of this IPR.

3. Specifically, I have been asked to provide my opinions regarding whether claims 1-3, 5, 7, 9-11, and 13 (each a "Challenged Claim" and collectively the "Challenged Claims") of the '424 patent would have been obvious to a person having ordinary skill in the art ("POSITA") as of October 28, 2015 (the earliest alleged priority date) and July 21, 2024. It is my opinion that each Challenged Claim would have been obvious to a POSITA after reviewing the prior art discussed herein.

II. BACKGROUND AND QUALIFICATIONS

4. I am an expert in the fields of computer science and information technology. In formulating my opinions, I have relied upon my training, knowledge, and experience in the relevant art. A copy of my curriculum vitae is appended to this declaration as Appendix A and provides a description of my professional experience,

including my academic and employment history, publications, conference participation, awards and honors, and more. The following is a brief summary of my relevant qualifications and professional experience.

5. I am a Professor in the Department of Computer Science at Boston University, and the Chair of Academic Affairs for the Faculty of Computing & Data Sciences. From 2013 to 2018 I also served as Chair of the Department of Computer Science. A brief summary of my background is as follows.

6. I earned an undergraduate degree in Biology from Cornell University, which I received in 1982. I received a master's degree in Computer Science from the University of Buffalo in 1989. I received a Ph.D. in Computer Science from the University of Rochester in 1994. The subject of my Ph.D. thesis was "Performance Prediction and Tuning of Parallel Programs."

7. From 1982 to 1984, I worked as a computer programmer for the State of Colorado. From 1984 to 1994, I was employed at Calspan Corporation, a research and development firm in Buffalo, NY, where I rose to the level of Senior Computer Scientist. My work at Calspan focused on development of experimental software and large-scale simulation software in support of contracts between Calspan and the U.S. Department of Defense.

8. In 1994, I joined the faculty of Boston University, initially as an Assistant Professor of Computer Science. I was promoted to the rank of Associate

Professor in 2000 and became a full Professor in 2006. I served as Chair of the Department of Computer Science from 2013 to 2018.

9. I have been pursuing research in the area of computer networking and data science since 1994. I have conducted research in a variety of areas related to the Internet and the World Wide Web, mainly through the application of data mining techniques and statistics. Among other areas, I have studied the efficient design of Web servers and content distribution systems. My research in network systems and data science includes research in the field of database and query systems, which are the backbone of services provided over the Internet and the World Wide Web. I have a published a number of research papers on topics related to distributed servers and datamining. My work in these areas is highly cited. From 2007 to 2009, I was the Chair of ACM SIGCOMM (the Special Interest Group in Computer Communication), the main professional organization for scientists in the field of computer networking.

10. I also spent time in industry working on topics related to network services and databases. I cofounded a startup (Commonwealth Network Technologies) which was acquired by Webmanage, Inc., where I had the title of Chief Scientist. At Webmanage, I designed content distribution systems. Webmanage was subsequently acquired by Network Appliance, Inc., as part of their Netcache division, where I became a Technical Director. At Network Appliance, I

helped design systems to distribute Web content and populate Web caches. I also served as Chief Scientist for the data science startup Guavus, developing big data solutions for the telecommunications industry.

11. I am co-author of Internet Measurement: Infrastructure, Traffic, and Applications (Wiley Press, 2006), and I am the author of over two hundred research papers. According to Google Scholar, my work has been cited by over 32,000 other publications. According to Google Scholar, as of May 2018, only 11 other researchers worldwide are more cited in the area of computer systems. I hold nine U.S. Patents derived from my Internet related research. I am a past editor of principal journals in related fields: Computer Communication Review, IEEE/ACM Transactions on Networking, Computer Networks, and IEEE Transactions on Computers.

III. MATERIALS CONSIDERED AND INFORMATION RELIED UPON REGARDING '424 PATENT

12. In preparing this declaration, I have reviewed the following materials bearing Exhibit Nos. that I understand are being referenced in the IPR to which my declaration accompanies:

LIST OF EXHIBITS

No.	Description
1001	U.S. Patent No. 12,143,424 ("**'424 patent**")
1002	File History of U.S. Patent No. 12,143,424
1003	N/A

No.	Description
1004	U.S. Patent No. 8,732,300 ("**Barsness**")
1005	U.S. Patent Application No. 2016/0269247 A1 ("**Chakradhar**")
1006	U.S. Provisional Application No. 62/132,686 ("**Chakradhar Provisional**")
1007	U.S. Patent Application Pub. No. 2014/0059185 A1 ("**Siripurapu**")
1008	Thomas Heinze, *et al*., *Cloud-based Data Stream Processing*, Proc. of the 8th ACM Int'l Conf. on Distributed Event-Based Sys., 238-245 (May 2014), https://di.acm.org/doi/10.1145/2611286.2611309 ("**Heinze**")
1009	Ashkon Soroudi, *et al*., *High Availability on a Distributed Real Time Processing System*, Electrical Engineering and Computer Sciences, University of California at Berkeley (May 15, 2015) ("**Soroudi**")
1010	Naresh Kumar Reddy Rapolu, *Dynamic Re-Optimization Techniques for Stream Processing Engines and Object Stores* (2015) (Ph.D. dissertation, Purdue University) ("**Rapolu**")
1011	Ankit Toshniwal, *et al*., *Storm @Twitter*, SIGMOD '14: Proc. of the 2014 ACM SIGMOD Int'l Conf. on Mgmt. of Data, 147-156 (2014), http://dx.doi.org/10.1145/2588555.2595641 ("**Toshniwal**")
1012	Leonardo Neumeyer, *et al*., *S4: Distributed Stream Computing Platform*, IEEE Int'l Conf. on Data Mining Workshops, 170-177 (2010), https://ieeexplore.ieee.org/document/5693297 ("**Neumeyer**")
1013	Michael Duller, et al., *Virtualizing Stream Processing*, Middleware (2011) ("**Duller**")
1014	Scott Schneider, et al., *Tutorial: Stream Processing Optimizations*, DEBS '13 (2013) ("**Schneider**")
1015	U.S. Patent No. 8,407,360 ("**Andrade**")
1016	U.S. Patent No. 8,560,527 ("**Santosuosso**")
1017	U.S. Patent No. 8,190,960 ("**Bahadur**")
1018	List of Challenged Claims
1019	N/A
1020	N/A

IV. UNDERSTANDING OF PATENT LAW

13. I am not an attorney. For the purposes of this declaration, I have been informed about certain aspects of the law that are relevant to my opinions. My understanding of the law was provided to me by the Petitioner's attorneys.

14. I understand that when considering the scope of the claims of a patent that the patent claim terms should generally be given the ordinary meaning that the terms would have to a person of ordinary skill in the art in question after reading the patent as of the earliest claimed priority date. As discussed below, it is my opinion that the challenged claims are not entitled to the earliest priority date asserted by the '424 patent. However, in my opinion, the meaning of the claim terms would not have materially differed to a person of ordinary skill in the art ("POSITA") whether the claims were evaluated at the time of the earliest claimed priority date (October 28, 2015) or at the time the application that led to the '424 patent was filed (July 21, 2024).

15. I understand that the person of ordinary skill in the art is deemed to read the claim term not only in the context of the particular claim in which the term appears, but in the context of the entire patent, including the other claims and the specification of the application as filed. I further understand that the principal considerations regarding the scope and meaning of the claims are the plain language of the claim (including the surrounding claim language and context), the patent

specification, and the prosecution history. I understand that while a claim is to be read in light of the specification, one must generally avoid importing limitations into the claim from the specification. I am also informed that the prosecution history can often inform the meaning of the claim by demonstrating how the inventor understood the invention and whether the inventor limited the invention in the course of prosecution, making the claim scope narrower than it would otherwise be. I applied these understandings when considering the scope and meaning of the claims of the ’424 patent.

16. I understand that a prior art reference anticipates an asserted claim, and thus renders the claim unpatentable, if all elements of the claim are disclosed in that prior art reference, either explicitly or inherently (i.e., necessarily present or implied).

17. I further understand that a claim is unpatentable if it would have been obvious. Obviousness of a claim requires that the claim would have been obvious from the perspective of a POSITA at the time the alleged invention was made. I understand that a claim could have been obvious from a single prior art reference or from a combination of two or more prior art references. I understand that an obviousness analysis requires an understanding of the scope and content of the prior art, any differences between the alleged invention and the prior art, and the level of ordinary skill in evaluating the pertinent art.

18. I further understand that a claim would have been obvious if it unites old elements with no change to their respective functions, or merely substitutes one element for another known in the field, and that combination yields predictable results. While it may be helpful to identify a reason for this combination, I understand that there is no strict requirement of finding an express teaching, suggestion, or motivation to combine within the references. When a product is available, design incentives and other market forces can prompt variations of it, either in the same field or different one. If a POSITA can implement a predictable variation, obviousness likely bars its patentability. For the same reason, if a technique has been used to improve one device and a POSITA would recognize that it would improve similar devices in the same way, using the technique would have been obvious. I understand that a claim would have been obvious if common sense directs one of ordinary skill, without hindsight bias caused by knowledge of the challenged claims, to combine multiple prior art references or add missing features to reproduce the alleged invention recited in the claims.

19. I further understand that certain factors may support or rebut the obviousness of a claim. I understand that such secondary considerations include, among other things, commercial success of the patented invention, skepticism of those having ordinary skill in the art at the time of invention, unexpected results of the invention, any long-felt but unsolved need in the art that was satisfied by the

alleged invention, the failure of others to make the alleged invention, praise of the alleged invention by those having ordinary skill or expertise in the art, and copying of the alleged invention by others in the field. I understand that there must be a nexus—a connection—between any such secondary considerations and the alleged invention. I also understand that contemporaneous and independent invention by others is a secondary consideration tending to show obviousness.

20. I am not aware of any allegations by the named inventors of the ’424 patent or any assignee of the ’424 patent that any secondary considerations tend to rebut the obviousness of any Challenged Claim of the ’424 patent.

21. I understand that in considering obviousness, it is important not to determine obviousness using the benefit of hindsight derived from the descriptions found in the patent being considered.

22. I understand that Petitioner has the burden of proving unpatentability by a preponderance of evidence, which means that the claims are more likely than not unpatentable.

23. The analysis in this declaration is in accordance with the above-stated legal principles.

V. <u>SUMMARY OF OPINIONS</u>

24. In forming the opinions set forth herein, I have considered:

a) The documents listed above;

b) My education, knowledge, skills, and experience in the field of computer science and engineering; and

c) The level of skill of a person having ordinary skill in the art at the time of the alleged invention of the ’424 patent, namely July 21, 2024, as well as at the alleged priority date of the ’424 patent, namely October 28, 2015.

25. As I explain in further detail below, it is my professional and expert opinion that claims 1-3, 5, 7, 9-11, and 13 of the ’424 patent are obvious in view of the prior art.

VI. LEVEL OF SKILL IN THE ART, AND STATE OF THE ART

A. Person Of Ordinary Skill In The Art

26. I understand that in evaluating obviousness of the ’424 patent claims, the content of a patent or printed publication prior art should be interpreted the way a person of ordinary skill in the art would have interpreted the prior art as of the effective filing date of the challenged patent claims. I assume that the effective filing date is the earliest filing date of an alleged related application, namely October 28, 2015. My understanding of a person of ordinary skill in the art would not materially differ if the effective filing date were the date the application resulting in the ’424 patent was filed, namely July 21, 2024, at least because the knowledge and level of

skill of a person of ordinary skill in the art in 2024 would encompass the knowledge and skill of such person in 2015.

27. I understand that factors that may be considered in determining the level of ordinary skill in the art at the time of the effective filing date of the challenged patent include: (1) the educational level of the inventor; (2) the types of problems encountered in the art; (3) prior art solutions to those problems; (4) rapidity with which innovations are made; (5) sophistication of the technology; and (6) educational level of active workers in the field.

28. A person of ordinary skill in the art ("POSITA") in July 2024 or October 2015 would have had a bachelor's degree in computer science, computer engineering, electrical engineering, or an equivalent degree, with two to four years' experience designing, implementing, or otherwise working with distributed computing systems. More education, e.g., a Master's or Ph.D., would compensate for less work experience and vice versa.

29. A POSITA would have been familiar with common, then-existing technologies for distributed computing and cloud-based stream processing systems, as well as issues surrounding the design, implementation, and maintenance of such technologies. The state of the art below reflects certain technologies within the knowledge and skill of a POSITA.

B. State Of The Art

1. Distributed, Highly Scalable, Stream Processing Engines Were Commonly Used For Real-Time Data Processing

30. By 2015, distributed stream processing engines were already well-established tools for real-time analysis of continuous data streams. Contemporary literature such as Heinze explained that cloud-based data stream processing systems were designed for high-throughput, low-latency processing of unbounded streams, and that newer "third generation" systems were specifically built to be "highly scalable and robust towards faults." EX1008, 238. Heinze explained that cloud-based streaming engines were "designed to dynamically scale to hundreds of computing nodes and automatically cope with varying workloads." EX1008, 240. These systems provided parallel processing of data streams and distributed multiple data streams / queries across a cluster. EX1008, 240. These "elastic data stream processing systems" could "dynamically vary the number of processing nodes within the system based on the workload." EX1008, 241; *id.*, 238, 240 ("Elasticity describes the behavior of a system to dynamically scale in or out based on a changing workload.").

31. Other references confirm that these characteristics were standard features of practical stream processing systems. As early as 2010, there was already "a clear need for highly scalable stream computing solutions that can operate at high

data rates and process massive amounts of data." EX1012, 170. Neumeyer described S4 in 2010 as a "distributed, scalable" platform that "allow[ed] programmers to easily develop applications for processing continuous unbounded streams of data," and explained that S4's design was intended for "low latency" operation in "large clusters built with commodity hardware." EX1012, 170. Neumeyer further explained that production environments required "scalability (ability to add more servers to increase throughput with minimal effort) and high availability (ability to achieve continuous operation with no human intervention in the presence of system failures)." EX1012, 170. Stream processing engines such as Storm were designed to be scalable, resilient, extensible, efficient, and easy to administer. EX1011, 147. Toshniwal described Storm as a "real-time fault-tolerant and distributed stream data processing system" used at Twitter "at scale, and in real-time," and explained that Storm was specifically designed so operators could "add or remove nodes from the Storm cluster without disrupting existing data flows," while continuing to operate on "large clusters." EX1011, 147.

32. Other literature reflecting the knowledge of a POSITA recognized scalability through parallelization and scale-out as a core requirement of streaming systems. Schneider explained that handling "large volumes of live data in short periods of time" made "high throughput processing" a "critical requirement," which in turn required taking advantage of "multiple cores and/or host machines to achieve

scale." EX1014, 249; *see also* EX1011, 1 ("Many modern data processing environments require processing complex computation on streaming data in real-time."); EX1009, 4 ("The term -real-time processing in this context means that the system is able to respond within seconds or sub-second to requests, while 'distributed' means that it is running on multiple computers."). Duller similarly explained that a stream-processing application could be "scale[d] out" across nodes and that the tested application "linearly scale[d] up the load … with the number of nodes." EX1013, 283.

33. By 2015, various "distributed stream processing engines," such as Aurora, Borealis, TimeStream, and open-source systems such as S4, Samza, and Storm were available "[t]o process continuous streams of data at low latency." EX1010, 2, 12; EX1011, 147.

2. Stream Processing Engines Typically Used Directed Graphs To Organize And Manage Processing Data Streams

34. It was well known by 2015 to use directed graphs to represent and manage the flow of data through streaming applications: "Stream engines code the application workflow as a directed acyclic graph (DAG) of operators, referred to as a topology." EX1010, 2; *see also* EX1013, 271-273 (describing a "data processing model" comprising a "graph" of operators). Further, "[t]uples are processed in a pipelined fashion as they traverse through the topology." EX1010, 2. "A topology is

a directed graph where the vertices represent computation and the edges represent the data flow between the computation components." EX1011, 148 ("The basic Storm data processing architecture consists of *streams of tuples* flowing through *topologies*.").

35. Schneider explained that a "stream processing application is organized as a *graph*, formed by a set of *operators* connected to each other by *streams*," where a "stream is a series of *data items*." EX1014, 250.

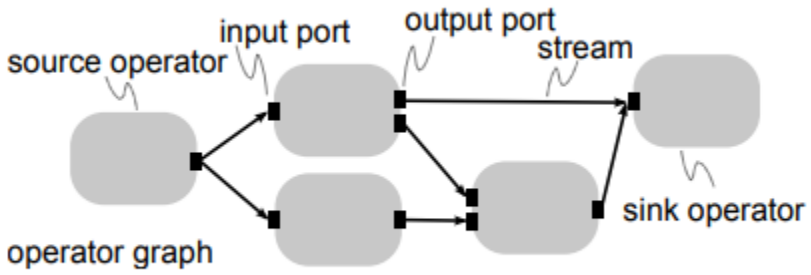


Figure 1: Basic concepts related to streaming applications.

EX1014, 250. Schneider explains how the operators of the graph are connected via input ports and output ports. *Id.* Further, source operators perform "edge adaptation to receive data from an external source," while sink operators perform "edge adaptation to deliver data" to an external sink. *Id.* Andrade, an IBM patent, similarly explains:

> A data-flow graph can include a set of processing elements (PEs) connected by streams, wherein each stream carries a series of stream data objects SDOs.) The PEs implement data stream analytics and are basic execution units that are

> distributed over the compute nodes. The PEs also communicate with each other via input and output ports, connected by streams.

EX1015, 5:31-41.

VII. THE '424 PATENT

A. The Claims' Earliest Effective Filing Date

36. For this declaration, I have been asked to assume that the claims' earliest possible effective filing date is October 28, 2015. That said, my opinions herein would not be materially different if the effective filing date was July 21, 2024.

B. The '424 Patent's Specification

37. The '424 patent describes the "analysis of very large data sets using distributed computational graph tools which allow for transformation of data through both linear and non-linear transformation pipelines." EX1001, 1:32-35. Among other things, it describes determining when aspects of the streaming data analysis "are processing slowly or may have hung … so that actions can be implemented to resolve the issue." EX1001, 16:41-56. The '424 patent does not, however, further describe these issue-resolving "actions" or how to implement them.

38. The '424 patent provides examples of prior art transformation pipelines (EX1001, FIGS. 6, 12) and other transformation pipelines (EX1001, FIGS. 7-9, 13-15). EX1001, 17:11-19:17, 21:25-22:51 (describing transformation pipelines).

However, the ’424 patent provides no disclosure of how to implement these transformation pipelines using the general hardware architecture shown in FIG. 3, with “any number of clients and/or servers.” EX1001, 13:50-62.

VIII. CLAIM CONSTRUCTION

39. For this declaration, I apply the plain and ordinary meaning of all claim terms, in addition to certain proposed constructions as noted below. I reserve the right to offer my opinions on any constructions or construction issues presented by Patent Owner.

40. I have been asked to apply the following proposed constructions:

“distributed computational graph” (claim 1): “*a directed graph with each transformation comprising a node and the output messages between transformations comprising edges*.” EX1001, 9:31-35; *see also id.*, 9:4 (“Definitions”); FIGS. 6-9 & 12-15.

“transformation” (claims 1-3, 5, 9-11): “a function performed on zero or more streams of input data which results in a single stream of output.” EX1001, 9:51-54.

“transformation pipeline” / **“pipeline”** (throughout the ’424 claims): “a progression of functions which each perform a transformation on a data stream.” EX1001, 4:31-34, 9:62-67.

IX. GROUND 1: CLAIMS 1-3, 5, 7, 9-11, AND 13 ARE OBVIOUS OVER BARSNESS

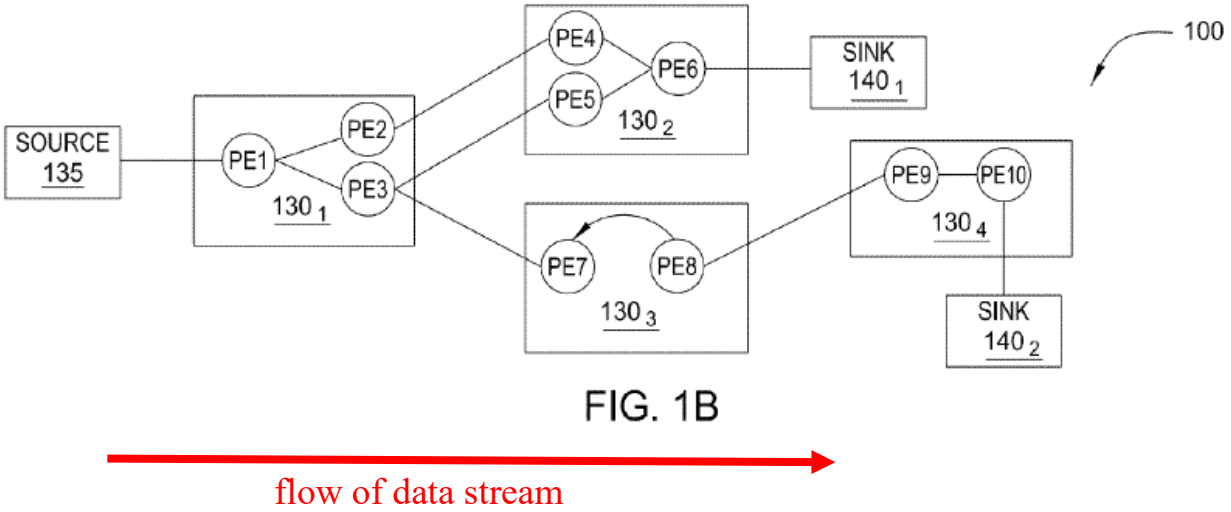
41. As explained below, claims 1-3, 5, 7, 9-11, and 13 are obvious over Barsness.

A. Barsness (EX1004)

42. Barsness is a U.S. patent that issued May 20, 2014, from an application filed January 10, 2011. EX1004, Cover.

43. Barsness discloses a distributed stream-processing computing cluster comprising a plurality of compute nodes executing processing elements ("PEs") that apply transformations to streaming data and transmit output data to downstream PEs according to an operator graph. Barsness explains that "[i]n a stream data processing application, operators are connected to one another such that data flows from one processing element (PE) to the next" and that "[s]calability is achieved by distributing an application across nodes." EX1004, 3:16-22.

44. Barsness further describes a computing infrastructure including a management system and multiple compute nodes connected via a network, where "the operator graph … represents a stream application beginning from one or more source processing elements (PEs) through to one or more sink PEs." EX1004, 6:43-56. Barsness depicts these data streams generally flowing left to right, e.g., FIG. 1B shows data flowing through PEs (1-10) on respective compute nodes ($130_{1\text{-}4}$)



Id., FIG. 1B (red annotations added).

45. As illustrated in FIG. 1B of Barsness, upstream processing elements collectively form a first transformation pipeline (e.g., 130_1) whose output is provided as input to downstream processing elements forming a second transformation pipeline (e.g., 130_2, 130_3, 130_4). Barsness describes that "data elements flow into a source PE of a stream application and are processed by that PE," and that output data elements subsequently flow to downstream PEs. EX1004, 6:49-56; 7:52-67.

46. Barsness teaches that routing of pipeline output messages is governed by topology information and performed at the operator level. Each compute node is a computer system comprising a processor, network interface, and memory. EX1004, 8:30-35. The memory stores processing elements and operators configured to process incoming data and to "emit data to other operators … and to other PEs in the stream application as described by the operator graph." EX1004, 8:66-9:20; 9:4-

9. Thus, Barsness requires that operators locally determine where to send output messages based on topology-related information.

47. Barsness further emphasizes the importance of dynamic reconfiguration in stream-processing systems. For example, Barsness teaches that a monitoring application may "increase the priority of the PE in question or even move the PE to separate machines to gain more resource," and that processing elements may be moved or replicated to improve performance and fault tolerance. EX1004, 3:56-60. Barsness also explains that the operator graph and compute nodes may be reconfigured by a user interface and management system. EX1004, 7:3-6; 10:28-39.

A. Claim 1

1. Element [1.1]

[1.1] A distributed computing cluster comprising:

48. Barsness satisfies the preamble. Barsness discloses a "computing infrastructure 100" that includes "a management system 105 and a plurality of compute nodes $130_{1\text{-}4}$, each connected to a communications network 120." EX1004, 6:43-48. This computing infrastructure is a *distributed computing cluster* because these separate computing elements communicate over the network to perform coordinated processing. FIG. 1A shows an example computing infrastructure and its distributed computing elements:

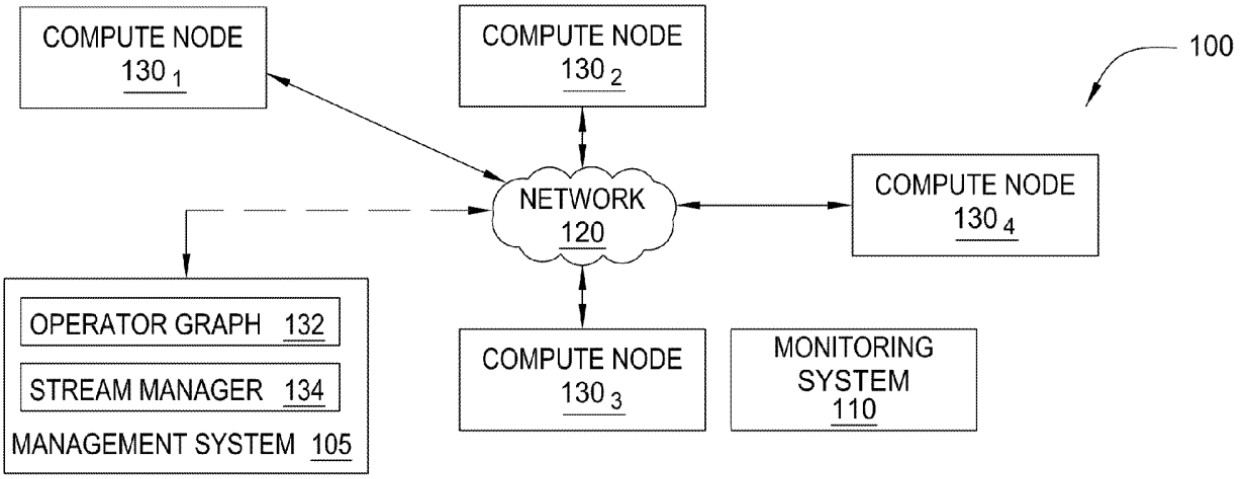


FIG. 1A

EX1004, FIG. 1A. Each compute node is "*a distributed* computing node." EX1004, 2:35-37 (emphasis added). Barsness also discloses that a monitoring system 110 of the computing infrastructure "may be a *cluster* of *a plurality of computing resources*, which may be *distributed* in the network 120." Thus, the computing infrastructure—with its "plurality of compute nodes" and "management" and "monitoring" systems distributed throughout network 120—is a *distributed computing cluster*.

49. A cluster was a well-known concept that referred to a group of machines working together. *E.g.*, EX1005, [0056] (executing Storm on a "cluster"); EX1007, [0349] (describing "hardware components including clusters of commodity nodes connected through commodity networking means," and the "MapReduce software framework supports distributed computing on large data sets on clusters of computers"), [0438]; EX1008, 241 ("multiple processing nodes within a cluster, which allows a distributed execution"); EX1009, 34 (discussing running

"Storm on a cluster of 8 virtual machines, each having 4 cores, 8GB of memory, and running CentOS 6"); EX1001, 10:32-35 (contrasting systems operating on "single machine" with systems "operating on a cluster of machines as is known in the art"). As explained in the following sections, the Barsness computing infrastructure comprises each of the recited features.

2. Element [1.2]

[1.2] a first plurality of computer systems,

50. Barsness satisfies this element. Barsness discloses that its computing infrastructure includes *a first plurality of computer systems*, e.g., management system 105 and "a plurality of compute nodes $130_{1\text{-}4}$." EX1004, 6:45-49. A first plurality of computer systems is shown in FIG. 1A of Barsness:

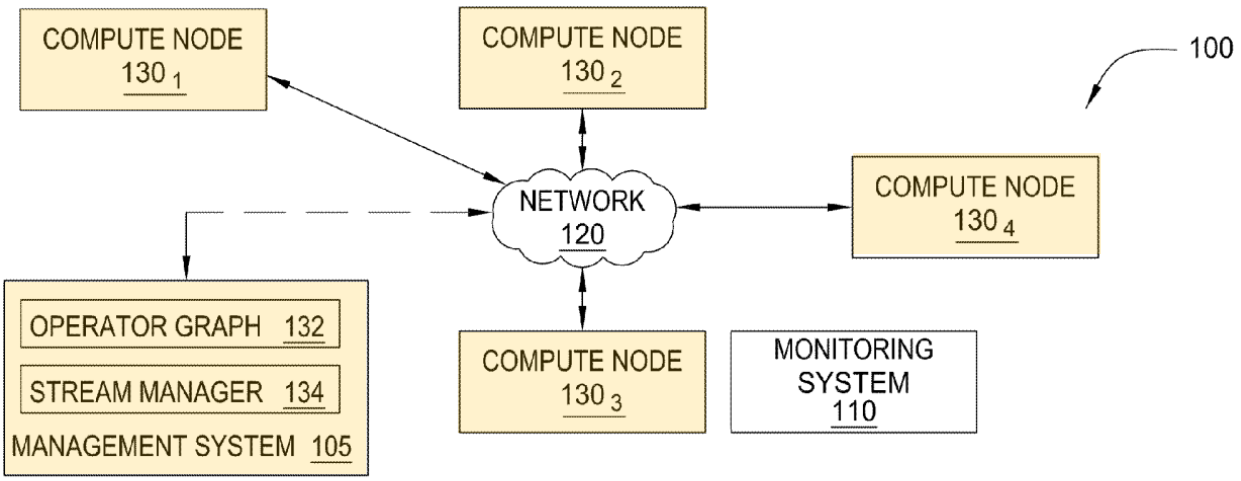


FIG. 1A

EX1004, FIG. 1A (highlighting added).

51. Each compute node is a *computer system* as it includes "a central processing unit (CPU) 205, a network interface 215, an interconnect 220, a memory 225, and storage 230," along with "an I/O devices interface 210 used to connect I/O devices 212)." EX1004, 8:30-39; *see also* EX1001, 14:63-15:23 ('424 patent describing exemplary "computer system" including basic elements such as "CPU" and "memory"). Barsness FIG. 2 shows details of a compute node:

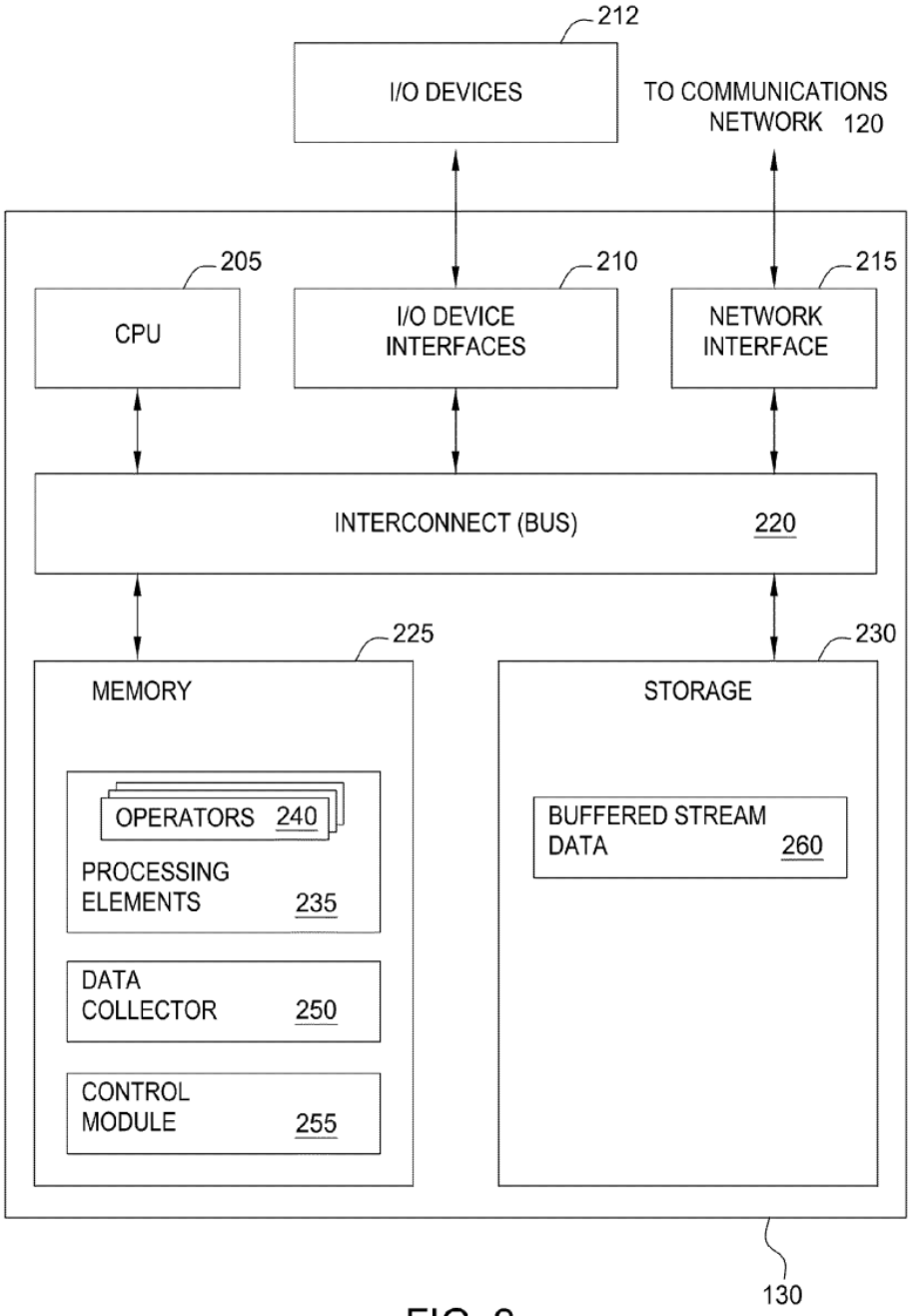


FIG. 2

EX1004, FIG. 2; *id.*, 8:53-65 (further describing the components of the compute node). Barsness discloses that "[t]he compute node 130 may be a physical host or a virtual machine," where "[a] virtual machine provides a software abstraction of

computing resources of a physical host computer" and "runs as an independent *computer system*." EX1004, 8:37-44 (emphasis added).

52. The management system 105 is also a *computer system* because it includes a "central processing unit (CPU), a network interface 315, an interconnect 320, a memory 325, and storage 330." EX1004, 9:53-59. Barsness FIG. 3 shows details of a management system:

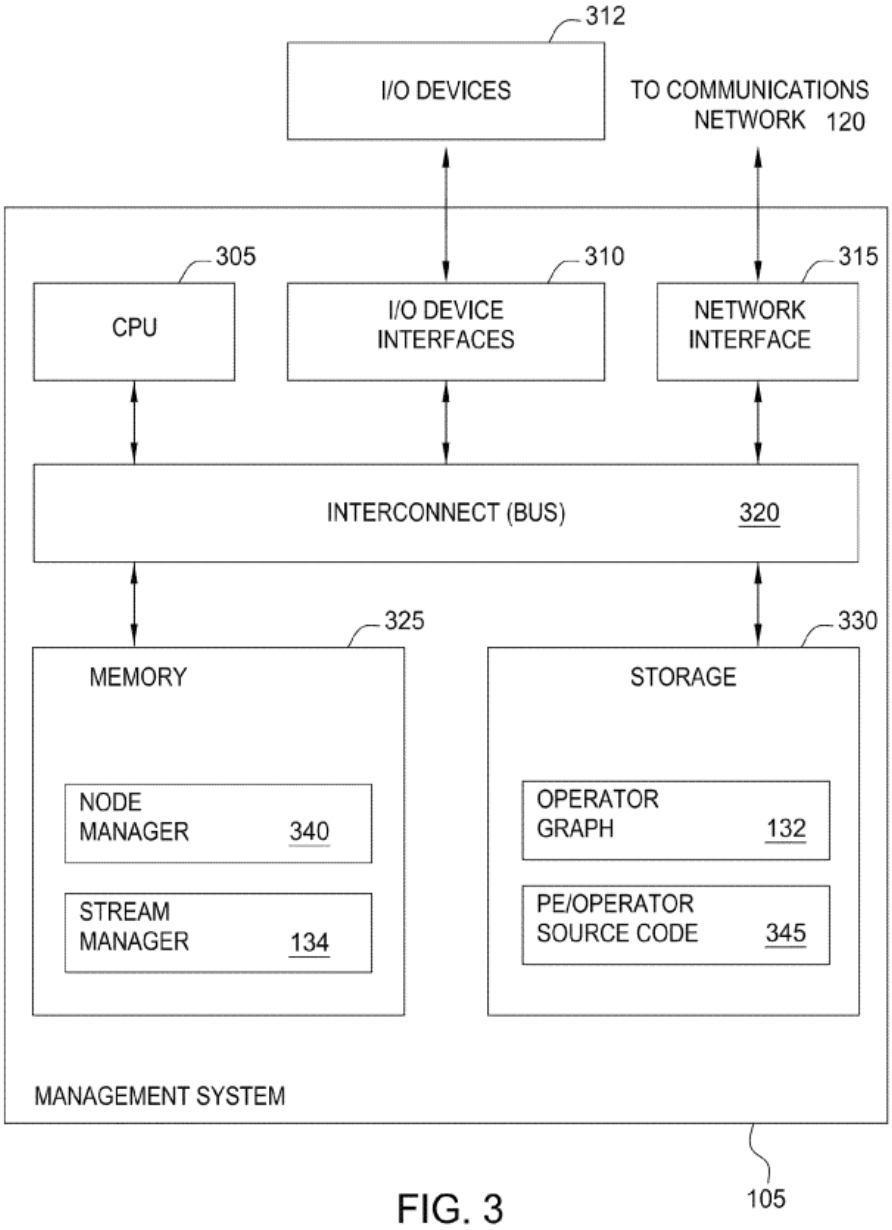


FIG. 3

EX1004, FIG. 3; *id.*, 9:60-10:64 (further describing the management system); *see also* EX1001, 14:63-15:23 ('424 patent describing an exemplary "computer system").

a) <u>Element [1.2.a]</u>

[1.2.a] wherein each respective computer system of the first plurality of computer systems comprises a memory that stores a respective first data,

53. Barsness satisfies this element. Each compute node (*respective computer system*) of the plurality of compute nodes (*the first plurality of computer systems*) in Barsness comprises "memory 225" (*a memory*) that "store[s]" data such as programming instructions and "application data" (*stores a respective first data*). EX1004, 8:53-55. The management system 105, also a *respective computer system of the first plurality of computer systems*, also comprises "memory 325" (*a memory*) that stores data, e.g., "programming instructions and application data." EX1004, 9:62-67. Each also includes a "storage" 230/330 that stores data, e.g., buffered stream data and operator graph 132. EX1004, 8:61-65, 10:3-11. The memory and storage (each alone and collectively, *a memory*) of a compute node and management system are depicted in FIGS. 2 and 3 of Barsness, as discussed below.

54. The "memory 225 is generally included to be representative of a random access memory," while "[s]torage 230, such as a hard disk drive, solid state device (SSD), or flash memory storage drive, may store non-volatile data." EX1004, 8:61-65; *id.*, 10:3-11 (similarly for the management system). The memory and storage of a compute node are depicted in FIG. 2 of Barsness:

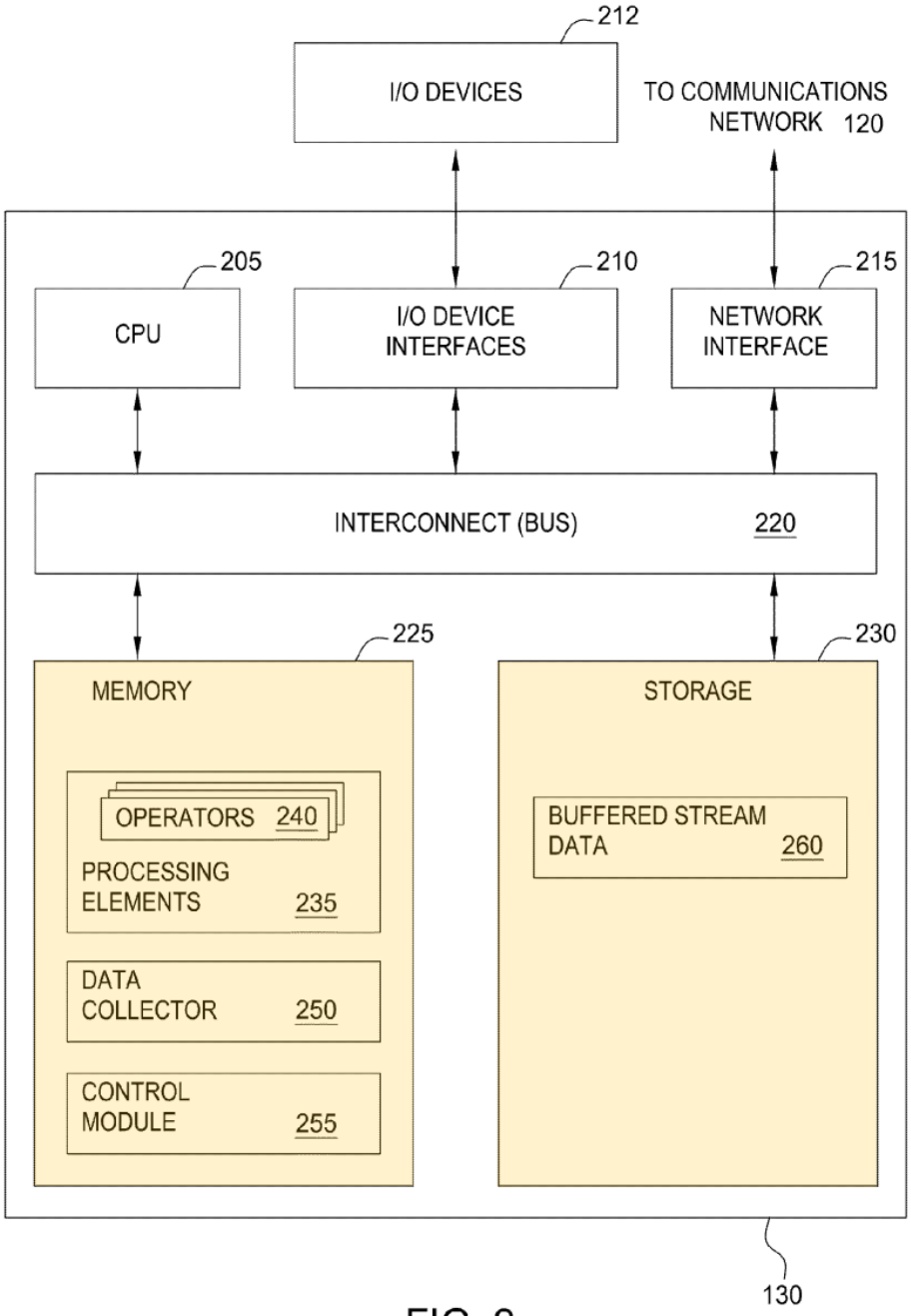


FIG. 2

EX1004, FIG. 2 (highlighting added).

55. The management system 105 (*respective computer system*) comprises memory 325 and storage 330 that stores, among other things, an operator graph 132 (*a respective first data*), as shown in FIG. 3 of Barsness:

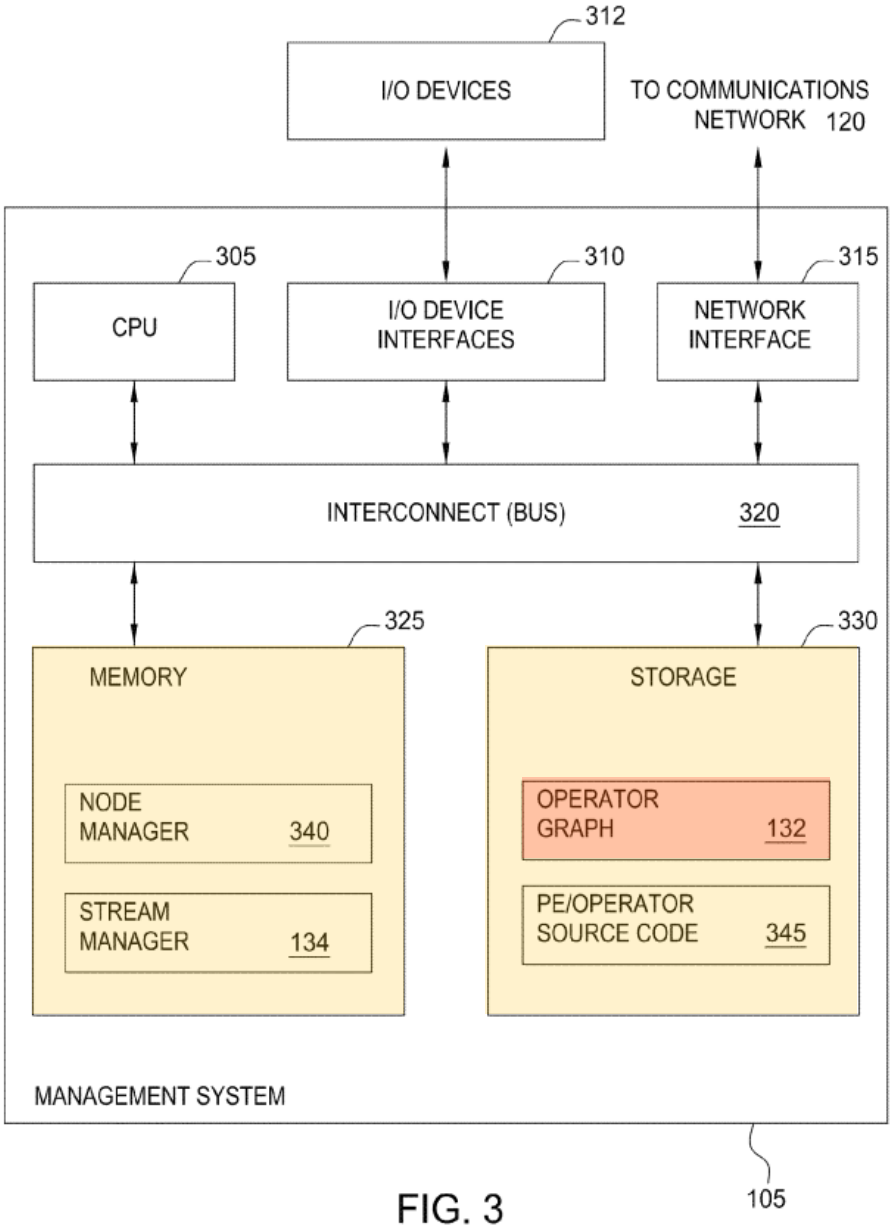


FIG. 3

EX1004, FIG. 3 (highlighting added); *id.*, 10:12-64.

56. As I explain in the next section, respective "data" stored in each respective memory of Barsness includes a *respective first data*.

b) Element [1.2.b]

[1.2.b] wherein the respective first data represents a respective portion of a distributed computational graph,

57. Barsness satisfies this element. Regarding a **distributed computational graph**, Barsness discloses an "operator graph" (*distributed computational graph*) that, for example, "represents the plurality of processing elements, and links between processing elements, corresponding to a flow of data tuples through the stream application." EX1004, 1:43-46; *id.*, 2:12-15, 2:59-62, 6:43-56, FIGS. 1A-1B; *id.*, 6:49-53("[T]he operator graph 132 represents a stream application beginning from of one or more source processing elements (PEs) through to one or more sink PEs."), 10:49-51 ("[T]he operator graph 132 maintains the connectivity configuration of a plurality of compute nodes and processing elements in a streaming application.").

58. FIG. 1B depicts an example operator graph, where the processing elements (PEs) are graph nodes and the links between processing elements are graph edges:[1]

[1] I am using the terms "graph nodes" and "graph edges" to distinguish these aspects of the operator graph from the "compute nodes" disclosed by Barsness.

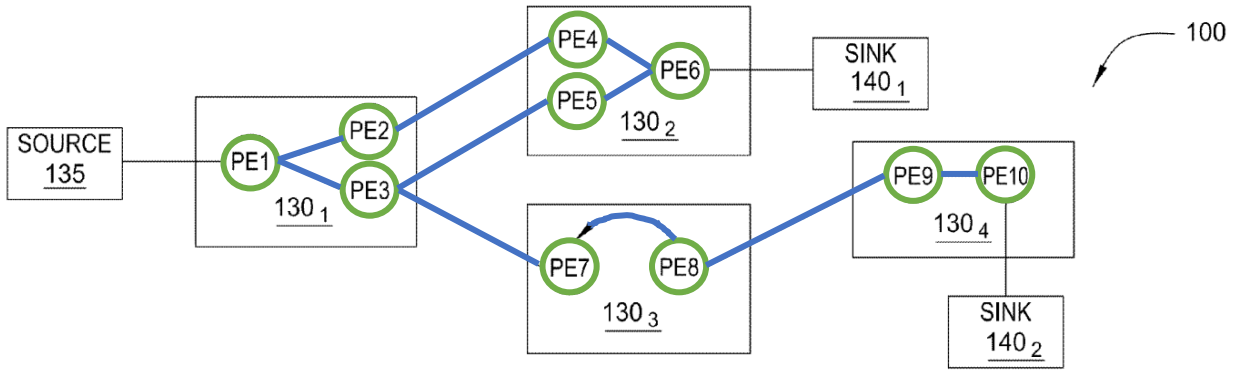


FIG. 1B

EX1004, FIG. 1B (green and blue annotations added). Processing elements are grouped into respective compute nodes 130 as shown. Each processing element includes an operator comprising executable code to perform an operation on data input to the processing element. EX1004, 3:16-35, 6:57-7:3 (explaining how "each processing element may be configured to carry out any form of data processing functions on a received N-tuple," which can include any "data analytic functions or operations"), 9:3-8, FIG. 2.

59. As noted above in paragraph 40, I have been asked to also construe "distributed computational graph" as a "directed graph with each transformation comprising a node and the output messages between transformations comprising edges." The operator graph of Barsness meets this construction because each graph node corresponds to a processing element (PE) that performs a transformation on data in the stream and each graph edge corresponds to the output from a given graph node (transformation) to the next graph node in the operator graph. More precisely:

- The operator graph is *directed* because the data stream flows in one direction from the source graph node to the sink graph node(s), with each graph edge indicating the flow of data output from one PE graph node to the next PE graph node. EX1004, 1:43-46, 2:13-15, 2:59-62, 6:43-56, FIG. 1B; *see also* EX1001, 9:27-31 ('424 patent explaining: "Under circumstances when a relationship flows from one node to another in one direction, … the order of the endpoints is significant. *Graphs with such edges are designated as 'directed.'*") (emphasis added). The arrow from PE8 back to PE7 depicted in Figure 1B confirms that the operator graph otherwise depicts the flow of data in a direction from left to right, from the source to the sink(s), in the figure.
- Each PE (graph node) comprises a *transformation* of data, i.e., "a function performed on zero or more streams of input data which results in a single stream of output." *See* paragraph 40. Each PE is configured to "carry out any form of data processing *functions*" including "data analytic *functions* or operations" on data received through the data stream. EX1004, 6:49-7:3 (emphasis added). To that end, each PE includes an operator comprising "a small chunk of executable code configured to process data flowing into a processing element". *Id.*, 9:3-9; *id.*, 3:16-22. The function performed by each PE results in a single

stream of output emitted "to other PEs in the stream application." *Id.*, 9:4-9; *see also id.*, 9:43-50; 7:55-56 ("Source PE1 emits tuples received by PE2 and PE3."); *compare with* EX1001, 18:37-42 (describing node sending same "output stream to two transformational pipelines"). Barsness describes, alternatively, that a PE "may split data elements," thus confirming that absent such a split Barsness expects a single output stream. EX1004, 7:56-57. In that alternative example where the stream is split such that a portion of the stream is sent to one downstream PE while the remainder is sent to another downstream PE, together this bifurcated stream still comprises a single data stream output by a PE.

- The edges of the operator graph are "links between processing elements, corresponding to a flow of data tuples through the stream application," and thus comprise the output messages between processing elements / operators, where the data stream emitted from a processing element comprises output messages from that processing element. EX1004, 2:12-15. A message is a unit of information passed between two elements, and each "tuple" emitted by a processing element is a message, or an "output message" because it is data output by the processing element. *E.g.*, EX1004, 6:57-65 (explaining that input and output data can be "in formats other than an N-tuple (e.g., the

processing elements could exchange data marked up as XML documents)" and that "[o]f course, the N-tuple received by a processing element need not be the same N-tuple sent downstream"); *see also* EX1005, [0122] (describing "output messages" from an operator); EX1015, 12:11-14 (another IBM patent explaining that "streaming data items from one PE to another involves *message transmission*"). The '424 patent does not provide any meaningful explanation of "output messages," but instead uses the term to refer to data emitted from a transformation node. *E.g.*, EX1001, 17:17-19 (an "output message" is the "transformed" output of a transformation node).

The operator graph of Barsness is further described below in paragraphs 64-68.

60. Under the plain and ordinary meaning of distributed computational graph, the operator graph is a *distributed computational graph* because, as I just explained, it is a graph that describes the distributed computations performed on a data stream, including the distribution of PEs across different compute nodes and the flow of data between PEs.

61. Within each compute node, memory stores **the respective first data [that] represents a respective portion of a distributed computational graph**. Barsness teaches that each compute node stores in its memory data corresponding to processing elements 235 and operators 240 associated with that compute node. For

example, "memory 225 includes one or more processing elements (PEs) 235." EX1004, 8:66-9:1; *id.*, FIG. 2. As each PE is a node in the operator graph, in storing a given compute node's respective PEs, each compute node memory thus stores data representing a respective portion of the operator graph. Barsness further explains that: "PEs 235 includes a collection of operators 240. Each operator 240 may provide a small chunk of executable code configured to process data flowing into a processing element." *Id.*, 9:3-12; *see also id.*, 9:13-33 (alternative manners of distributing operators and PEs across compute nodes); *id.*, 3:19-22 ("Scalability is achieved by distributing an application across nodes by creating many small executable pieces of code (operators), as well as replicating processing elements on multiple nodes and load balancing among them."), 2:3-4 ("The compute nodes are configured to execute processing elements of a stream application."), 1:39-41 ("stream application composed from a plurality of processing elements executing on one or more compute nodes"), 1:55-57 (same). Source code for the processing element and operator is loaded by the management system into the compute node to setup the processing elements and operators. EX1004, 10:51-56 ("The processing element and operator source code 345 is loaded to setup the processing elements 235 and the operators 240 in the compute nodes 130."). The operator graph of FIG. 1B shows how first compute node 130_1 includes PE1, PE2, and PE3, and thus stores respective first data representing a respective portion of the operator graph. EX1004,

FIG. 1B. Similarly, the other compute nodes $130_{2\text{-}4}$ store respective processing elements as depicted in the operator graph.

62. The processing elements and operators stored in the memory 225 of the compute nodes are at least a part of the "programming instructions and application data" (*respective first data*) stored in the memory 225 of the compute nodes (*the first plurality of computer systems*) because the processing elements and operators comprise the stream application. EX1004, 8:53-55, 1:39-41, 2:3-4. Specifically, the operators stored in the memory of a compute node are *respective first data* because the operators are "executable pieces of code" that are each "configured to process data flowing into a processing element (e.g., PEs 235) and to emit data to other operators 240 in a PE and to other PEs in the stream application as described by the operator graph 132." EX1004, 3:19-22, 9:4-9. Within a given compute node's memory, the data for the PEs and operators *represents a respective portion of a distributed computational graph* because those PEs and related operators correspond to at least that portion of the Barsness operator graph (*distributed computational graph*) associated with the respective compute node. *E.g.*, EX1004, 2:12-15 ("The operator graph represents the plurality of processing elements, and links between processing elements"), 10:51-56 ("The processing element and operator source code 345 is loaded to setup the processing elements 235 and the operators 240 in the compute nodes 130."). Within the operator graph, each processing element (and its

related operator) corresponds to a graph node of the operator graph. Thus, the data representing the PEs / operators (*respective first data*), loaded and stored in the memory of each compute node, *represents a respective portion of an* operator graph (*distributed computational graph*).

63. The *respective portion of a distributed computational graph* stored in the memory (memory 325 and storage 330) of the management system 105 includes the full operator graph 132 (*respective first data*), which allows the management system 105 "to control a stream application running on the compute nodes $130_{1\text{-}4}$ as well as change the structure of the operator graph 132." EX1004, 7:27-29; *id.*, 6:48-56. By storing the operator graph 132 in the management system 105, the node manager 340 and stream manager 134 of the management system 105 can "mov[e] processing elements (PEs) from one compute node 130 to another, e.g., to manage the processing loads of the compute nodes 130," and otherwise build and configure the operator graph. EX1004, 7:27-37; *id.*, 10:12-64 (describing management of the operator graph by the management system).

c) <u>Element [1.2.c]</u>

[1.2.c] and wherein the distributed computational graph describes a flow of output data of a first transformation pipeline to an input of a second transformation pipeline,

64. Barsness satisfies this element. The operator graph of Barsness is a distributed computational graph, as I explained above in paragraphs 57-59. The

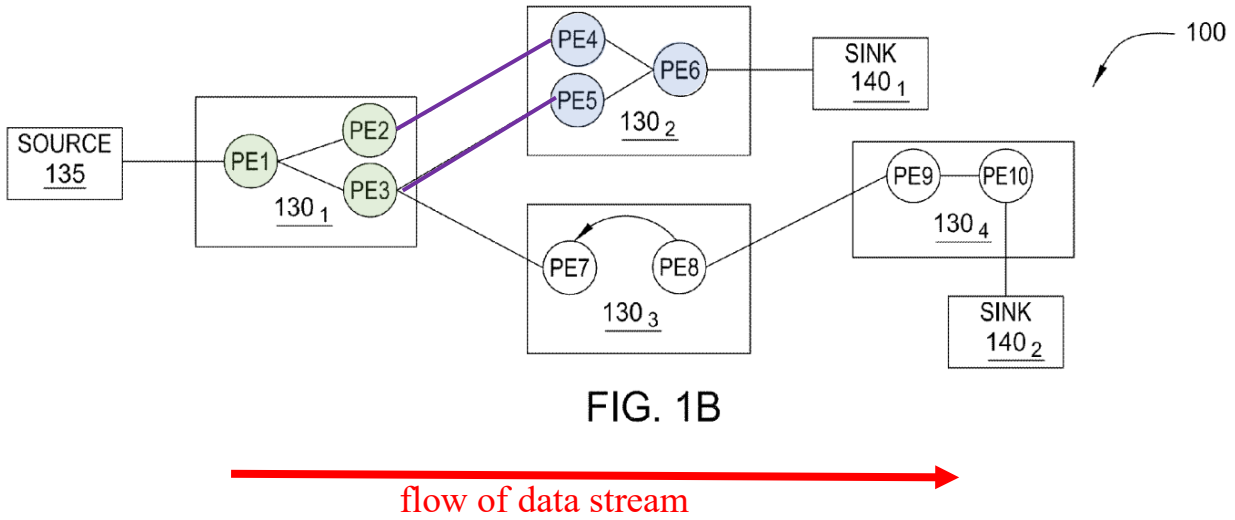
operator graph describes the flow of data through several compute nodes, each containing a group of processing elements (PEs) that process / transform the data flowing through them. *E.g.*, EX1004, 8:11-15 ("PEs may be configured to transform incoming data tuples by either adding or removing one or more data attributes before emitting outgoing data tuples to other PEs, as per the configuration of target PEs."); *see also id.*, 7:52-54.

65. Further, the group of PEs for a given compute node taught by Barsness satisfies a *transformation pipeline*, as construed above in paragraph 40, because those PEs form "a progression of functions which each perform a transformation on the data stream" passing through the compute node. In the Barsness stream data processing application, "operators are connected to one another such that data flows from one processing element (PE) to the next." *E.g.*, EX1004, 3:16-18; *see also id.*, 3:22-28. As the data flows from one PE to the next, each PE performs a data processing operation on the data via its operator and emits the transformed data to the next PE(s) in the graph. EX1004, 6:57-7:3; *id.*, 6:55-56 ("Data elements flow into a source PE of a stream application and are processed by that PE."), 3:16-22 (describing operators), 9:4-7 (explaining how PEs on a compute node include "a collection of operators" that "provide a small chunk of executable code configured to process data flowing into a processing element"); *see also* paragraph 59. Barsness teaches that each PE may have "only one" active operator. EX1004, 9:9-23. The

group of PEs within a given compute node thus comprise a progression of functions that each perform some action or transformation on the data stream.

66. For example, the operator graph depicted in Barsness FIG. 1B "begins at a source PE 135 (labeled as PE1) and ends at sink PEs $140_{1\text{-}2}$ (labeled as PE6 and PE10)." EX1004, 7:52-54. As shown in FIG. 1B, each compute node (boxes $130_{1\text{-}4}$) contains a group of PEs that form a *transformation pipeline*. Here, the progression of transformations performed by PE1, PE2, and PE3 in the compute node 130_1 is a *first transformation pipeline*. A pipeline can include branching and iterative (cyclical) configurations. *E.g.*, EX1001, 9:67-10:7.

67. Accordingly, as shown in annotated FIG. 1B below, the transformations performed by processing elements PE1, PE2, and PE3 form a first transformation pipeline, the transformations performed by processing elements PE4, PE5, and PE6 form a second transformation pipeline, and the operator graph describes a flow of output data from the first transformation pipeline to inputs (PE4 and PE5) of the second transformation pipeline, as shown by the graph edges between PE2-PE4 and PE3-PE5 respectively:



EX1004, FIG. 1B (highlighting and red annotations added); *id.*, 7:52-67 (describing the flow of data tuples through the nodes and processing elements of the example operator graph).

68. Other combinations of processing elements can be considered pipelines as well. For example, the combinations of processing elements PE1-PE2 and PE1-PE3 are each a transformation pipeline for the same reason PE1-PE2-PE3 is a transformation pipeline. Similarly, PE4-PE5 and PE4-PE6 are transformation pipelines for the same reason PE4-PE5-PE6 is a transformation pipeline. These examples show Barsness operator graph implementations that describe data streams flowing through a series of interconnected transformation pipelines. The processing elements PE7 and PE8 in the third compute node 130_3, where data flows from PE7 to PE8 and then loops back to PE7 (e.g., to perform iterative or cyclical processing before passing to PE9), are also a (second) transformation pipeline. EX1004, 7:64-

67. The processing elements PE9 and PE10 in the fourth compute node 1304 are also a transformation pipeline.

69. Under the plain and ordinary meaning of transformation pipeline, the group of PEs for a given compute node is a *transformation pipeline* because they form a pipeline of transformations, where each PE performs a transformation and the group of PEs forms a series of transformations on a data stream. EX1004, 6:57-7:3; *id.*, 6:55-56, 3:16-22, 9:4-7.

3. Element [1.3]

[1.3] wherein a first computer system of the first plurality of computer systems is configured to:

70. Barsness satisfies this element. For example, the first plurality of computer systems includes compute node 130_1, which is *a first computer system of the first plurality of computer systems*. EX1004, 7:54-55, 8:30-34. The first compute node (*first computer system*) within this plurality of systems is depicted in FIG. 1B of Barsness:

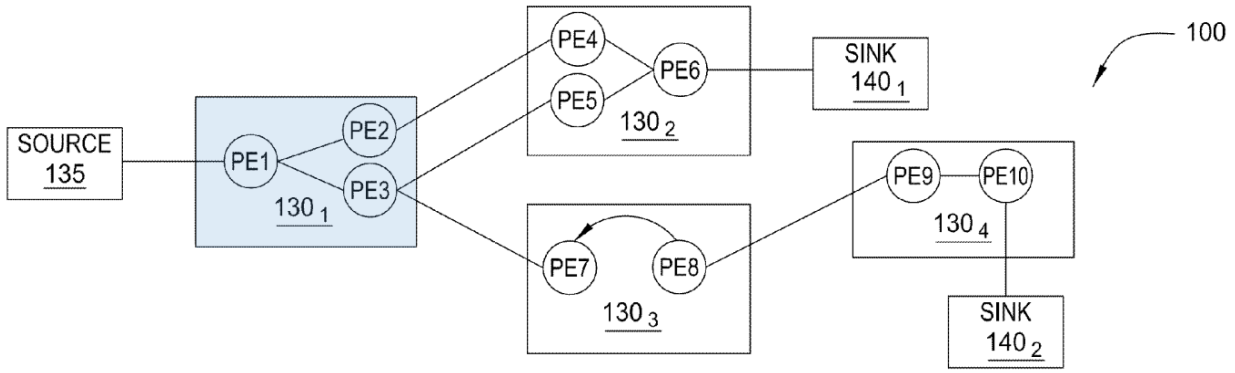


FIG. 1B

EX1004, FIG. 1B (highlighting added); *see also id.*, FIG. 2 (depicting a compute node), 8:30-34 (explaining that "the compute node 130 includes, without limitation, a central processing unit (CPU) 205, a network interface 215, an interconnect 220, a memory 225, and storage 230").

a) <u>Element [1.3.a]</u>

[1.3.a] receive a first stream of input data from a first input feed,

71. Barsness satisfies this element. Barsness discloses "[d]ata elements flow[ing] into a source PE of a stream application" (*a first stream of input data from a first input feed*). EX1004, 6:49-56; *id.*, 7:52-54. Further, Barsness discloses "a data source for the stream application" (*first input feed*) and "a storage space for data flowing into the compute node." EX1004, 9:43-46. This is shown in FIG. 2, where storage stores "[b]uffered stream data 260":

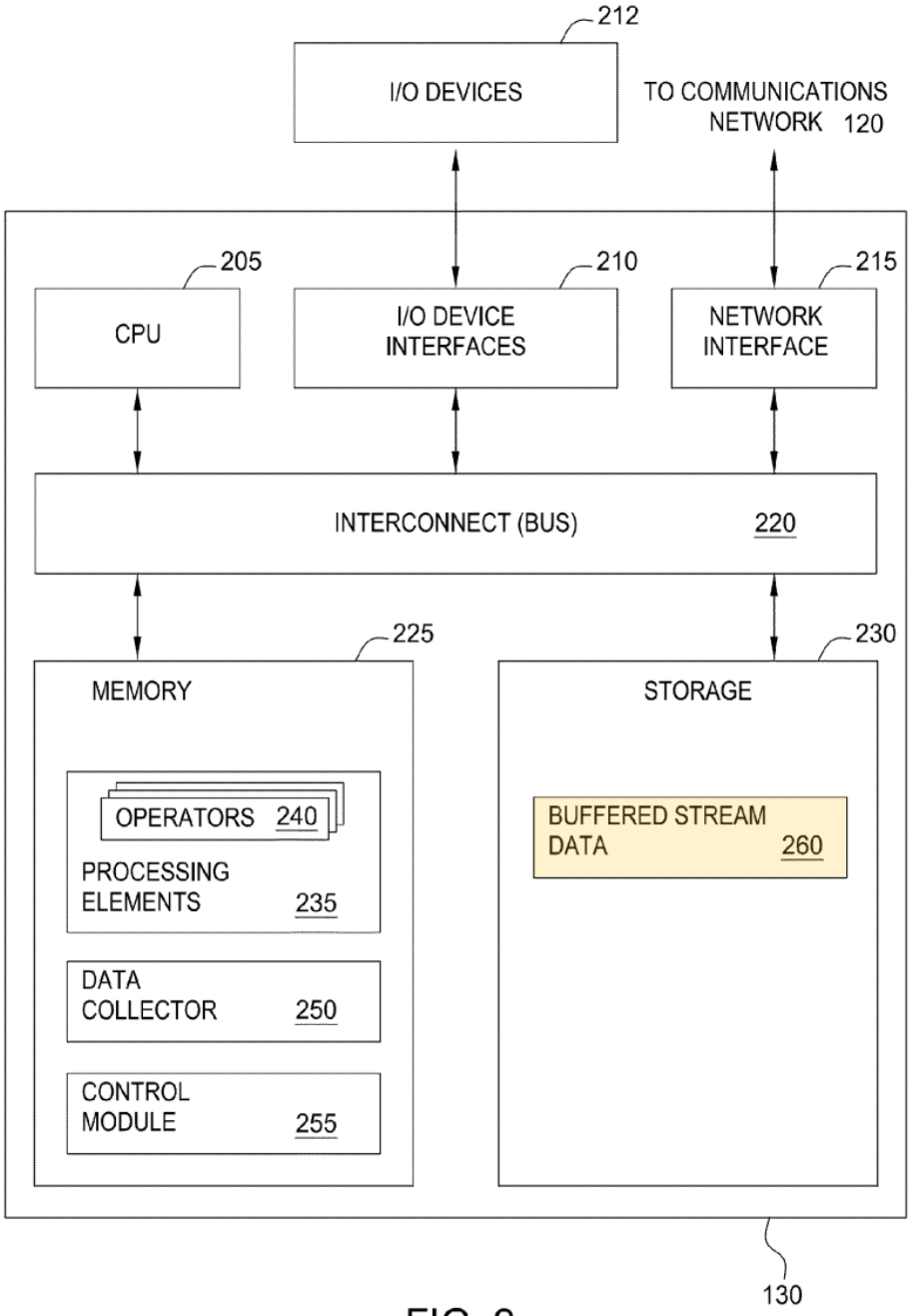


FIG. 2

EX1004, FIG. 2 (highlighting added).

72. Barsness discloses that the "buffered stream data may include data tuples waiting to be processed by one of the PEs 235." EX1004, 9:43-47; *id.*, 9:47-

50. A POSITA would have understood that the "buffered stream data" is the data elements received as an incoming stream from an upstream source, because Barsness repeatedly refers to "streaming data" or "stream data" being processed by processing elements of a stream application. *E.g.*, EX1004, 10:40-48 ("The stream manager 134 may be used to monitor the *data stream* as it flows through the processing elements in a stream application.") (emphasis added); *id.*, 1:22-24, 3:16-18, 9:43-50, 10:20-26.

73. Further, the first compute node 130_1 (*first computer system*) includes "a source PE 135 (labeled as PE1)" (EX1004, 7:52-54) that *receives* the "[d]ata elements flow[ing] into a source PE of a stream application," as depicted in FIG. 1B:

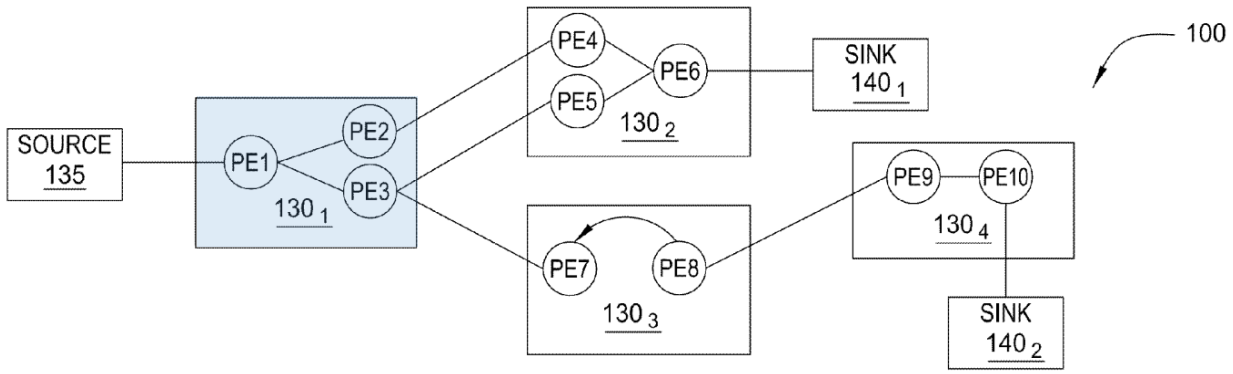


FIG. 1B

EX1004, FIG. 1B (highlighting added), 6:49-56.

b) <u>Element [1.3.b]</u>

[1.3.b] process the first stream of input data substantially in real time by executing software instructions that apply the first transformation pipeline to the first stream of input data to generate first pipeline output messages,

74. Barsness satisfies this element. As I explained for elements 1.2.b through 1.2.c, each compute node processes the source streaming data using transformation pipelines comprising processing elements. *See* paragraphs 57-68. For example, the first compute node 130_1 (*first computer system*) of Barsness *processes the first stream of input data* from the source 135 *by applying a first transformation pipeline* via a plurality of processing elements, with processing elements PE1, PE2, and PE3 forming a first transformation pipeline applied to the data elements received from the first stream of input data at PE1:

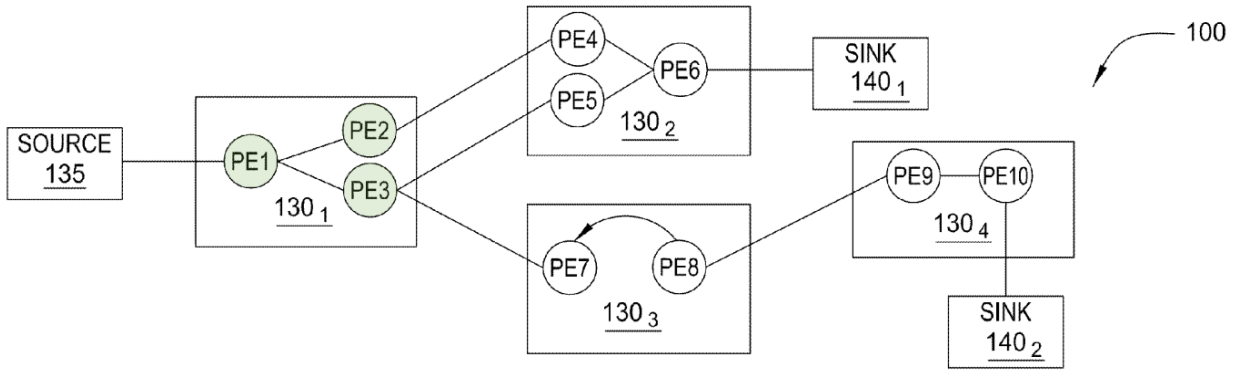


FIG. 1B

EX1004, FIG. 1B (highlighting added).

75. Regarding **executing software instructions that apply the first transformation pipeline**, Barsness teaches that a CPU of a given compute node / computing system, including compute node 130_1, "retrieves and executes programming instructions stored in the memory 225," the memory storing one or more processing elements, and each processing element including an operator.

EX1004, 8:53-54, 8:66-9:14; *see also id.*, 7:13-15 ("load programming instructions and configurations to be executed by the compute nodes $130_{1\text{-}4}$"). Each processing element performs "data analytic functions or operations" according to the operator provided at the processing element. EX1004, 6:65-7:3; *id.*, 7:40-44 ("a processing element may be executed as an independently running process"). Further, each operator is a "small chunk of executable code configured to process data flowing into a processing element (e.g., PEs 235) and to emit data to other operators 240 in a PE and to other PEs in the stream application as described by the operator graph 132." EX1004, 9:4-9; *id.*, 3:19-22 ("small executable pieces of code (operators)"). The operators of processing elements in a compute node are executed by a CPU of the compute node, as shown in FIG. 2 of Barsness:

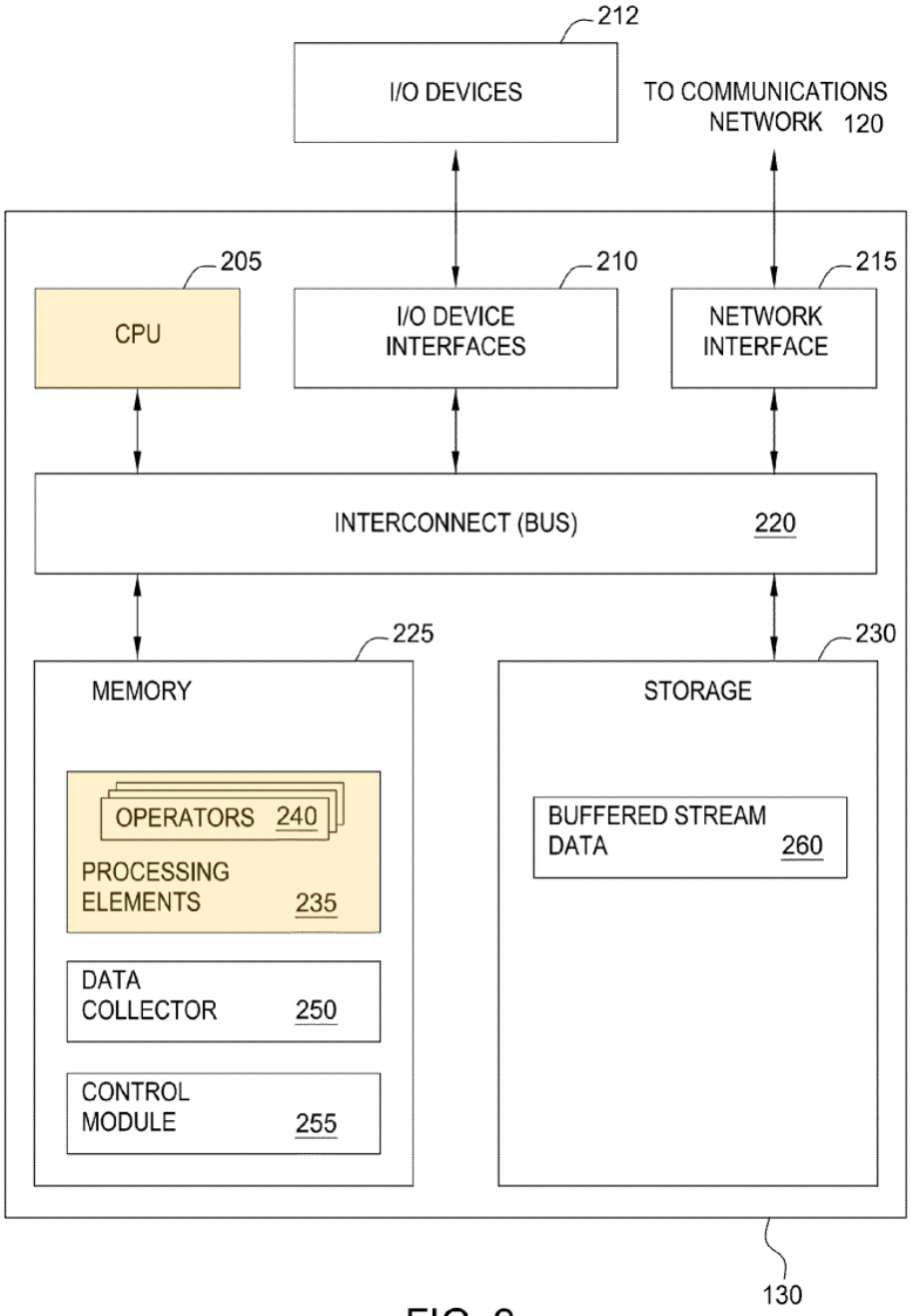


FIG. 2

EX1004, FIG. 2 (highlighting added). As the processing elements and operators in the compute node form the first transformation pipeline as described by the operator graph, executing the software instructions comprising the operators thus applies the first transformation pipeline to the incoming data stream.

76. Regarding **first pipeline output messages**, Barsness discloses that the "processing elements … *emit an N-tuple of data elements into the stream* (except for a sink PE where the stream terminates)." EX1004, 6:57-60 (emphasis added). The output data emitted by the processing elements PE2 and PE3 are *first pipeline output messages* because they are the resulting output of the progression of transformations applied to the input data by the processing elements PE1, PE2, and PE3 (*the first transformation pipeline*):

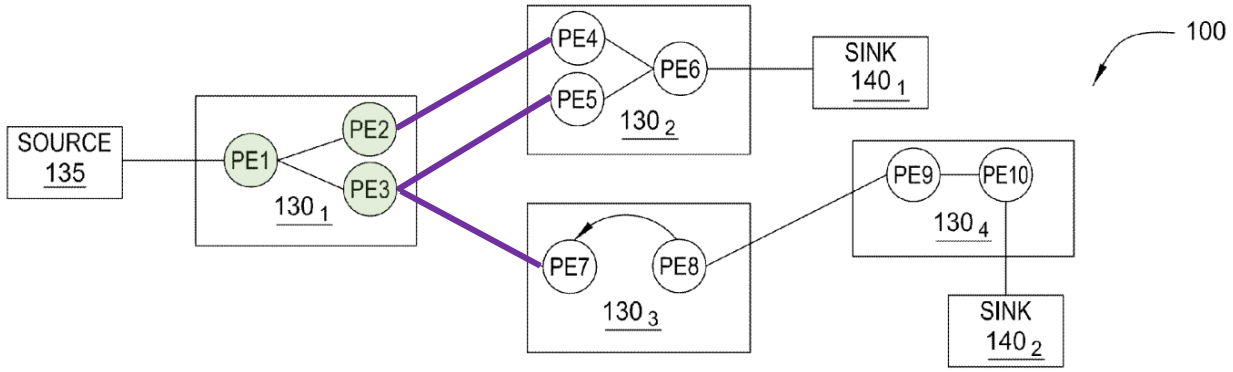


FIG. 1B

EX1004, FIG. 1B (highlighting added); *id.*, 6:62-65 (PEs "receive or emit data in formats other than an N-tuple"), 7:52-67 (describing the flow of data tuples through the nodes and processing elements of the example operator graph); *see also* paragraphs 64-68 (element 1.2.c).

77. Barsness teaches processing the data stream in real-time. EX1004, 2:52-3:15 (describing "real-time monitoring"). Barsness describes its "stream data processing application" system in the context of providing improvements to "real-

time data processing," i.e., "when the incoming data need to be processed in real-time," and "without losing the real-time characteristics of the data processing." EX1004, 2:58-61, 3:3-6, 3:16-18. Barsness further describes "processing logic to apply to inbound data records while they are 'in flight,' with the results available in milliseconds." EX1004, 1:24-27.

78. The architecture of Barsness confirms that it teaches real-time data processing. For example, Barsness explains that database systems are slow for the needs of real-time data processing. EX1004, 1:15-19 ("For example, the rate at which a database system can receive and store incoming data can limit how much data can be processed or otherwise evaluated, which, in turn, limits the utility of database applications configured to process large amounts of data in real-time."). In contrast, Barsness teaches an architecture configured such that input data streams proceed directly to processing, rather than first being stored in a database, with the data stream provided directly to source PE 135 of the first compute node 130_1. EX1004, 7:52-56, FIG. 1B.

79. Further, the monitoring application of Barsness monitors "current tuple processing rates at a selected PE" so that "the monitoring application 435 may act and reconfigure the processing modules of the stream application" when the throughput of a particular processing element slows down, to maintain real-time processing. EX1004, 12:28-36; *id.*, 3:61-4:2 (describing cloning a PE "that may be

running too slowly to process data tuples through the stream"), 12:40-13:3 (discussing monitoring processing rates of a selected processing element, with example of "receiving 100 tuples per second at the input but only producing 30 tuples at the output" potentially indicating "processing issues at the selected processing element" where the element historically processed 70 tuples per second), 13:12-14 (disclosing "employ[ing] corrective measures automatically when a discrepancy in tuple processing rates is found"), 13:24-29 (describing adding an operator to maintain minimum processing speed), 14:30-39 (describing "corrective measures" taken to maintain the tuple processing rate, which "include adjusting the run-time priority of a particular processing job in the stream application to bring the current tuple processing rates within the pre-configured range"), FIG. 6 (at 612, "employing corrective measures in the stream application according to pre-set configurations"). These dynamic pipeline adjustments, which are used to ensure that the data stream is processed with sufficient throughput, further show that the compute nodes, including the first compute node 130_1 (*the first computer system*), *process the input data stream substantially in real time*. EX1004, 3:3-6.

c) <u>Element [1.3.c]</u>

[1.3.c] process the respective first data stored in the memory of the first computer system to determine information about the second transformation pipeline,

80. Barsness satisfies this element. As I explained above in paragraphs 53-63 (elements 1.2.a-1.2.b), Barsness discloses storing first data representing a respective portion of an operator graph (*distributed computational graph*) in the memory of each respective compute node (each a *computer system*), including the first compute node 130_1 (*the first computer system*).

81. For example, Barsness teaches that each operator is "executable code configured to process data flowing into a processing element (e.g., PEs 235) and to emit data to other operators 240 in a PE and to other PEs in the stream application as described by the operator graph 132." EX1004, 9:4-9, FIG. 2 (showing operator in memory of compute node). Similarly, Barsness describes a PE as passing data elements on to subsequent PEs. *Id.*, 7:56-58 ("PE1 may … pass some data elements to PE2 others to PE3"); *see also id.*, 8:14-15 (describing PEs "configured to" transform data before "emitting outgoing data tuples to other PEs, as per the configuration of target PEs."). Barsness further describes a PE in one compute node emitting data to a next PE on another compute node. *Id.*, 7:58-59 ("Data that flows to PE2 results in tuples emitted to PE4 on compute node 130_2."); *id.*, 7:60-61 ("data tuples flowing from PE3 to PE5"). The operator graph "maintains the *connectivity configuration* of a plurality of compute nodes and processing elements in a streaming application." EX1004, 10:49-51 (emphasis added); *id.*, 2:12-15 (operator graph

represents the "links between processing elements, corresponding to a flow of data tuples through the stream application"), 3:45-50.

82. Barsness describes PEs using TCP/IP sockets to direct output data to the next PE(s): "operators are connected to one another such that data flows from one processing element (PE) to the next (e.g., over a TCP/IP socket)." EX1004, 3:16-18; *id.*, 3:24-28. Barsness provides a specific example where "inter-process communication" between two processing elements "occur[s] using a network socket (e.g., a TCP/IP socket)." EX1004, 7:44-47. POSITAs knew that TCP/IP communications included a port (socket) number and IP address to allow precise communication among devices, as referenced by Barsness. *E.g.*, EX1004, 9:9-12 ("configured to receive data traffic at a[n] incoming port of the PE"), 11:42-45 ("each input and output port of each operators 240 in a processing element (PE)"). POSITAs further knew that this involved each device (here, each PE) storing TCP/IP socket information for its target PE(s), in order to transmit output data to those PE(s).

83. Bahadur reflects background knowledge regarding TCP/IP. Bahadur describes implementing inter-process communication (IPC) with the Transmission Control Protocol (TCP) operating above the Internet Protocol (IP) network layer protocol, and explains that "[w]hen a TCP connection is established between two applications, TCP/IP provides a number of services for the packet communications between the two applications," such as "reorder[ing] packets received out of order,

automatically transmit[ting] lost packets in a stream, prevent[ing] packet duplication, check[ing] for transmission errors, and implement[ing] flow control procedures." EX1017, 1:22-42. Bahadur explains that inter-process communication between two applications using a TCP/IP connection involves "specification of a 'socket' that comprises the IP address associated with the remote processor and the port number associated with the application running on that processor." EX1017, 1:43-50. Further, "[w]hen a packet arrives at the TCP layer on a processor, the specific TCP connection is identified by the combination of a source socket (the source IP address and source port) and the destination socket (the destination IP address and destination port)." EX1017, 1:50-54. Setting up a network socket between two applications using a TCP/IP socket allows transmission of packets between the applications, where a TCP connection is established and maintained between a port bound to one application and a port bound to the other application. EX1017, 1:55-2:26, FIG. 1. Given this background knowledge, POSITAs would have understood from the Barsness disclosure of processing elements communicating with each other "using a network socket (e.g., a TCP/IP socket)" (EX1004, 7:44-47) as disclosure that the compute nodes store such connection information (e.g., IP address and port number) to connect processing elements with each other.

84. As further confirmation, Santosuosso (EX1016, another IBM patent) with similar disclosure of the operator graph and a management system for processing streaming data expressly teaches each compute node storing stream connection data. EX1016, FIG. 2. Specifically: "The stream connection data 255 represents the connections between PEs on compute node 130 (e.g., a TCP/IP socket connection between two separate PEs 235), as well as connections to other compute nodes 130 with upstream and or downstream PEs in the stream application, also via TCP/IP sockets (or other inter-process data communication mechanisms)." EX1016, 8:56-62. This confirms that a POSITA would have understood that Barsness, through its disclosure of using TCP/IP for communication between processing elements, discloses each compute node storing TCP/IP socket connection information to enable communication between PEs (including between PEs in different compute nodes).

85. Thus, Barness discloses the application data stored in the first compute node 130_1 (*the respective first data stored in the memory of the first computer system*) includes an identification of the specific "incoming port" for downstream PE(s) (*information about the second transformation pipeline*), and the first compute node 130_1 (*first computer system*) processes this application data to obtain the port number for routing output data to the downstream PE(s). EX1004, 9:9-12.

86. For example, in the example operator graph disclosed by Barsness, the processing elements PE2 and PE3 of the first compute element 130_1 / first transformation pipeline emit output data to processing elements PE4 and PE5 of the second compute element 130_2 / second transformation pipeline:

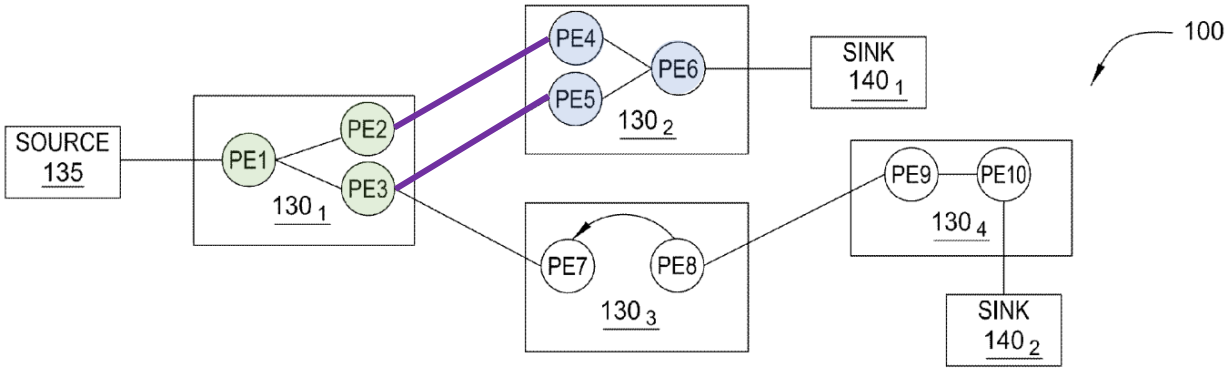


FIG. 1B

EX1004, FIG. 1B (highlighting added); *id.*, 7:52-67 (describing the flow of data tuples through the nodes and processing elements of the example operator graph). In order for the processing elements PE2 and PE3 of the first transformation pipeline to direct the output data to the processing elements PE4 and PE5 of the second transformation pipeline, the respective operator of each processing element PE2 and PE3 stored in the memory of the first computer system, includes information identifying the processing elements PE4 and PE5 as a destination (*information about the second transformation pipeline*). For example, using TCP/IP, first compute node 130_1 stores socket information including the "source socket (the source IP address and source port) and the destination socket (the destination IP address and

destination port)" (EX1017, 1:43-54), which would include the destination socket information for the downstream PEs PE4 and PE5.

87. The first compute node thus executes the operator of a given processing element at the end of the first transformation pipeline (e.g., PEs 2 and 3 in example operator graph of Barsness FIG. 1B) (*processes the respective first data stored in the memory of the first computer system*) to determine the destination of its output tuples (*to determine information about the second transformation pipeline*).

88. A POSITA would thus understand that the first PE is configured with (and therefore knows, at least logically) which downstream PE(s)/port(s) are the targets for its emitted data. At the very least a POSITA implementing Barsness would have been motivated to configure the operator / executable code of each PE with information identifying the downstream PE(s) to which the given operator should emit its output data, such that the PE emitted its transformed data to the appropriate subsequent PE(s) in the stream application. A POSITA would have been motivated to implement the PEs in such a way so that they could emit the data to destination(s) without having to consult the operator graph stored separately on the management system. Having this information on each PE would allow for rapid transmission of outputs to the next downstream node.

d) Element [1.3.d]

[1.3.d] and transmit the first pipeline output messages to a second computer system of the first plurality of computer systems in accordance with the determined information,

89. Barsness satisfies this element. Barsness discloses a given computing node transmitting outgoing data to a second computing system of the first plurality of computer systems per the configuration represented by the operator graph. EX1004, 8:11-15. In the example provided by Barsness, the processing elements PE2 and PE3 in the first compute node 130_1 emit/transmit data (*transmit the first pipeline output messages*) to PEs 4 and 5 of the second compute system 130_2 (*a second computer system of the first plurality of computer systems*):

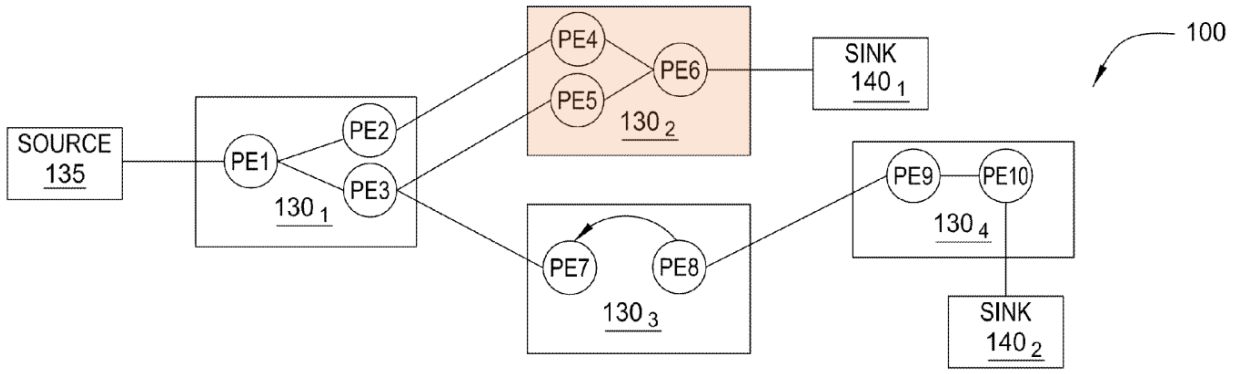


FIG. 1B

EX1004, FIG. 1B (highlighting added); *supra* paragraphs 64-68. In FIG. 1B, this transmission is reflected in each of the edges between PE2-PE4 and PE3-PE5. As discussed above in paragraphs 80-88, the operators of Barsness identify the destination of their output data, and thus the first compute node (*first computer*

system) transmits the output of the first compute node (*the first pipeline output messages*) to the second computer system according to the destination(s) specified in the operator (*the determined information*). EX1004, 3:16-18 (inter-PE communication via TCP/IP socket), 7:44-47.

4. Element [1.4]

[1.4] wherein the second computer system is configured to:

90. Barsness satisfies this element. Barsness discloses a second compute node 130_2 (*second computer system*) as shown in FIG. 1B:

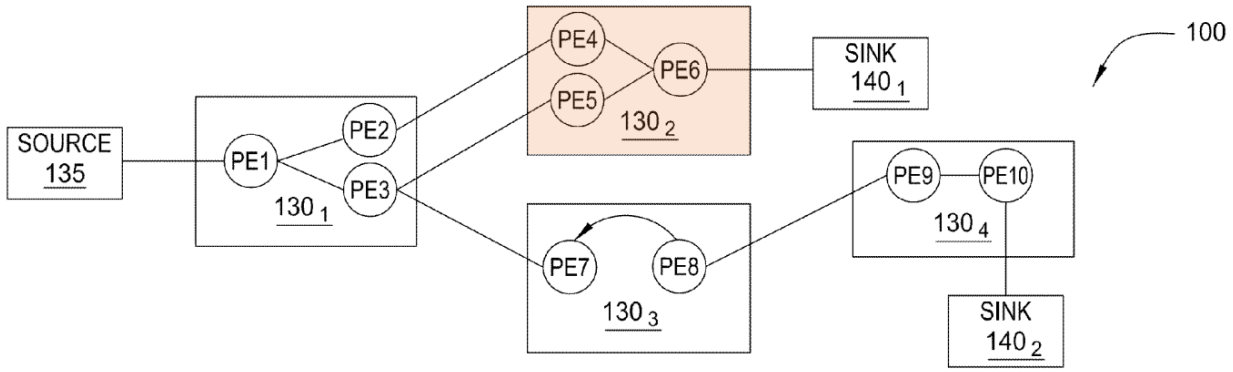


FIG. 1B

EX1004, FIG. 1B (highlighting added). The following sections explain why Barsness teaches configuring this compute node in the manner recited by claim 1. The Barsness third compute node 130_3 is an alternative "second computer system" with a real-time transformation pipeline of PE7 and PE8 that transmits output messages to PE9, for the same reasons explained below for compute node 130_2. For example, the pipeline comprising the processing elements PE7 and PE8 may

alternatively comprise the second transformation pipeline because they are downstream from the processing elements of the first compute node 130_1 comprising the first transformation pipeline and process the output of that pipeline (i.e., the output of PE3, first pipeline output messages). Further, the processing element PE8 of the third compute node 130_3 transmits its output (pipeline output messages) to PE9 of the fourth compute node 130_4.

a) <u>Element [1.4.a]</u>

[1.4.a] receive the first pipeline output messages,

91. Barsness satisfies this element. Barsness discloses the second compute node 130_2 (*the second computer system*) *receiving* the data emitted by PEs of the first compute node 130_1 (*the first pipeline output messages*). EX1004, 7:58-61, FIG. 1B. For example, as discussed above in paragraphs 64-68, the operator graph of Barsness shows a flow of output data (*the first pipeline output messages*) from the first transformation pipeline to inputs (processing elements PE4 and PE5) of the second transformation pipeline in the second compute node 130_2 depicted in FIG. 1B:

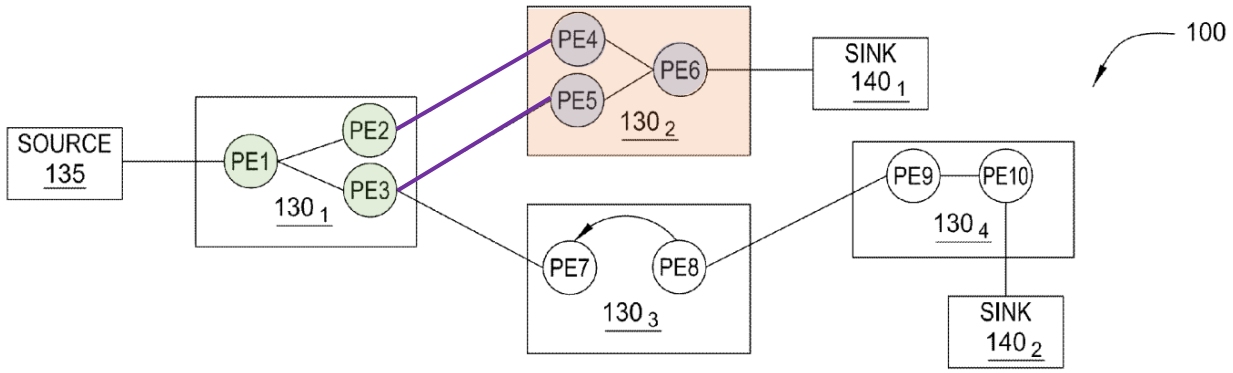


FIG. 1B

EX1004, FIG. 1B (highlighting added). Barsness further describes "[d]ata that flows to PE2 results in tuples emitted to PE4 on compute node 130_2," and similarly "data tuples flowing from PE3 to PE5," where PE5 is also on compute node 130_2. EX1004, 7:58-61. Barsness describes the processing elements transmitting the messages "using a network socket (e.g., a TCP/IP socket)." EX1004, 7:44-47.

b) <u>Element [1.4.b]</u>

[1.4.b] and process the first pipeline output messages substantially in real time by executing software instructions that apply the second transformation pipeline to the first pipeline output messages to generate second pipeline output messages,

92. Barsness satisfies this element. For example, in FIG. 1B, the second compute node 130_2 (*second computer system*) processes the output data from the first compute node 130_1 (*the first pipeline output messages*) by executing the operators / executable code (*executing software instructions*) associated with processing elements PE4, PE5, and PE6 (*the second transformation pipeline*) to generate output data (*second pipeline output messages*):

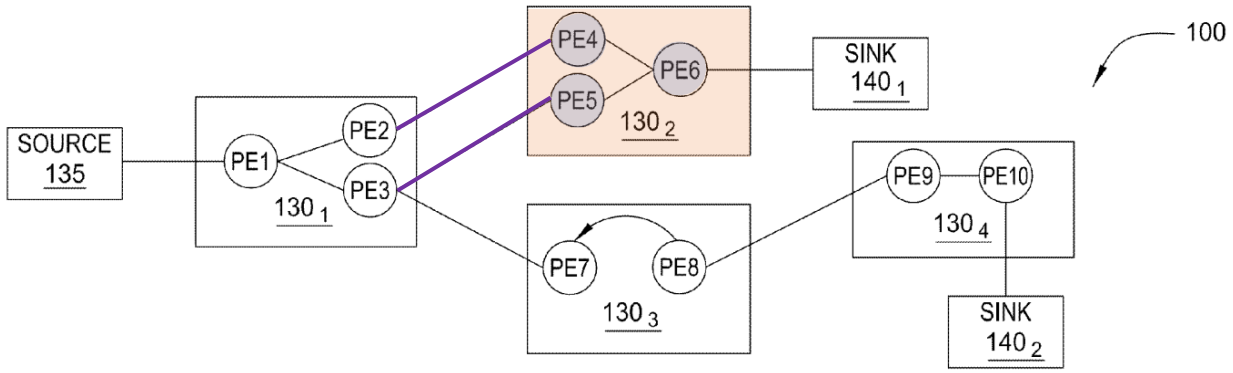


FIG. 1B

EX1004, FIG. 1B (highlighting added); *see also id.*, 7:52-67. Barsness further teaches configuring sink PEs to send data to other applications (*generate second pipeline output messages*). *Id.*, 8:4-6 ("Sinks 140 may also be configured to send data tuples to other applications in the computing infrastructure 100."). POSITAs knew how to output data from sink PEs using edge adaptation, as I discussed above in the state of the art. *See* paragraph 35. Further, a POSITA would have been motivated with a reasonable expectation of success to implement a given sink PE to generate output messages to other applications that used the data at least because Barsness teaches doing so. Thus, in the FIG. 1B example, PE6 (and thus the *second transformation pipeline*) generates output data that are *second pipeline output messages* because PE6 is the last PE in the second transformation pipeline, and so its output messages are output messages of the second transformation pipeline. And as I mentioned above, compute node 130_3, an alternative "second computing

system" with transformation pipeline PE7 and PE8, also generates output messages to PE9 of compute node 130_4.

93. The PEs within a given compute node apply a transformation pipeline by processing the stream of input data. EX1004, 8:11-15 ("In one or more embodiments, PEs may be configured to transform incoming data tuples by either adding or removing one or more data attributes before emitting outgoing data tuples to other PEs, as per the configuration of target PEs"). This occurs in real time, as explained with respect to the PEs of compute node 130_1 (*first transformation pipeline*) and Element 1.3.b. *See* paragraphs 74-79.

5. <u>Element [1.5]</u>

[1.5] wherein the first and second computer systems are distinct; and

94. Barsness satisfies this element. The compute nodes of Barsness, including the first compute node 130_1 and the second compute node 130_2 are shown as distinct computing systems (as is the third compute node 130_3):

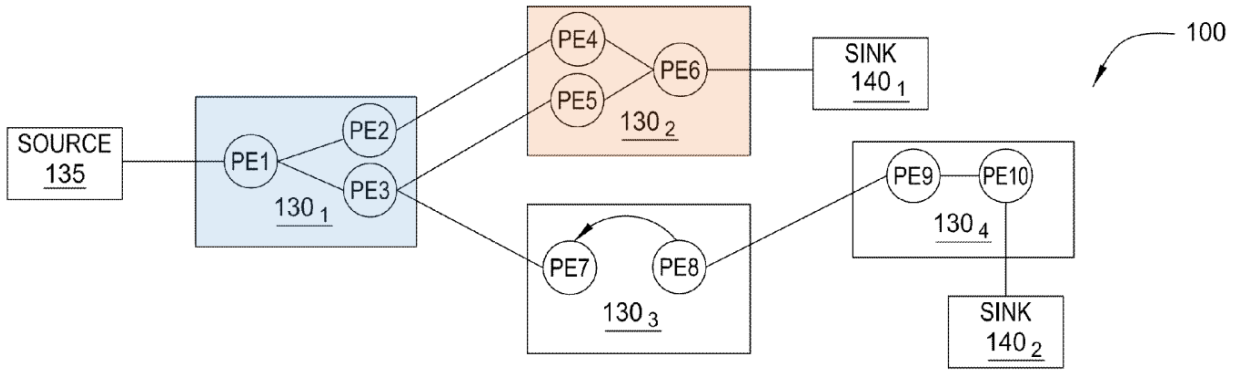


FIG. 1B

EX1004, FIG. 1B (highlighting added). When implemented as a virtual machine, each compute node "runs as an independent computer system." EX1004, 8:43-44.

95. Barsness further teaches moving PEs to "separate machines," further confirming that the compute nodes are distinct machines. EX1004, 3:56-60 ("In response, the monitoring application could then increase the priority of the PE in question or even move the PE to separate machines to gain more resource, assuming that the user responds favorably to prompts suggesting such changes to increase performance.").

6. Element [1.6]

[1.6] a second plurality of computer systems;

96. Barsness satisfies this element. Barsness discloses automatically adding additional compute nodes to the compute nodes $130_{1\text{-}4}$ to improve the processing of the stream application. EX1004, 14:44-56; *id.*, 14:26-39, 13:12-14, FIG. 6. POSITAs would thus understand that the system includes additional, unconfigured compute nodes available for use. Such available compute nodes, on their own or together with one or more of the compute nodes $130_{1\text{-}4}$ executing the stream application, are *a second plurality of computer systems*. The "second plurality of computer systems" is arbitrary, such that any group of two or more compute nodes satisfies it, including combinations of available / duplicated compute nodes and one or more of the

compute nodes $130_{1\text{-}4}$, multiple duplicated compute nodes, a duplicated compute node and an unprovisioned / available compute node, and so on.

97. Barsness discloses implementing its stream data processing application using a "cloud computing infrastructure," where "the nodes used to create a stream application may be virtual machines hosted by a cloud service provider." EX1004, 6:16-42. This infrastructure includes a "shared pool of configurable computing resources that can be rapidly provisioned and released." EX1004, 6:23-27. The infrastructure thus provides virtual computing resources, including "complete virtualized computing systems." EX1004, 6:26-31; *id.*, 6:32-40. Thus, as Barsness teaches that certain "virtual machines hosted by a cloud service provider" would serve as the compute nodes (EX1004, 6:40-42), POSITAs would understand that other hosted virtual machines remained available for use, even if not currently executing one or more processing elements. Two or more of these virtual machines that are available but not currently assigned to execute make up a *second plurality of computer systems*. EX1004, 6:24-26. Similarly, one or more of such virtual machines together with one or more of the in-use virtual machine (e.g., compute nodes $1301_{\text{-}4}$ executing the processing elements of the stream application) would also make up a *second plurality of computer systems*.

98. Specifically, Barsness discloses that each compute node "may be a physical host or a virtual machine," where "a physical host may be configured to

host a plurality of virtual machines, which may share hardware resources of the physical host" and where each "virtual machine runs as an *independent computer system*." EX1004, 8:37-44 (emphasis added). Barsness further discloses that the "processing nodes and the management server may be hosted in a virtualized computing environment to provide well known advantages of virtualized computing infrastructure including another *layer of scalability* and fault tolerance." EX1004, 8:48-52 (emphasis added). Through this "scalability and fault tolerance" of the virtualized computing infrastructure, the plurality of compute nodes taught by Barsness thus includes as many "independent computer system[s]" as needed for executing stream applications, including *a second plurality of computer systems*. EX1004, 8:43-44, 8:48-52. For example, Barsness discloses "spawn[ing] more process nodes 130" as needed. EX1004, 10:28-34; *id.*, 14:51-56 ("spawning new compute nodes" and "duplicating a compute node"), 8:45-48 ("spawn more virtual machines on the fly").

99. While I am referring to the virtualized cloud-computing embodiment disclosed by Barsness, Barsness also discloses that each compute node 130 may be a "physical host," so the teachings in Barsness regarding adding compute nodes 130 apply equally for a physical embodiment. EX1004, 8:30-39.

7. Element [1.7]

> [1.7] wherein a third computer system of the first plurality of computer systems is configured to execute software instructions that cause a fourth computer system of the second plurality of computer systems to execute software instructions that apply at least one of the first transformation pipeline and the second transformation pipeline.

100. Barsness satisfies this element. As explained in the following paragraphs, Barsness discloses the management system 105 (*third computer system*) configured to duplicate or migrate an existing compute node to an additional compute node (*fourth computer system*), thereby causing that additional compute node to apply the transformation pipeline of the duplicated or migrated compute node (*first / second transformation pipeline*). For example, with reference to FIG. 1B, Barsness teaches a management system configured to address performance issues detected in compute node 130_1 by deploying an additional compute node 130_x (*fourth computer system of the second plurality of computer systems*) that executes software instructions applying the same PEs of compute node 130_1 (*first transformation pipeline*):

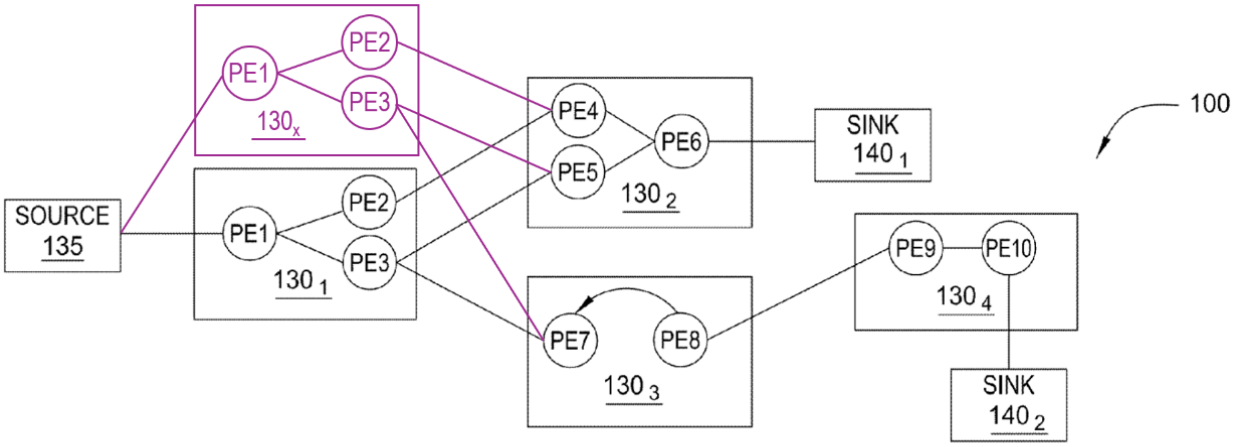


FIG. 1B

EX1004, FIG. 1B (purple annotations added).

101. Specifically, Barsness discloses its management system being configured to deploy additional computer systems to increase performance. For example, the management system will provision or "spawn more process nodes 130 to process certain type of data [to address] a detection of performance issues with a computer node 130." EX1004, 10:28-34. And it will "move processing elements 235 from one computer node 130 to another to improve the data processing performance of the stream application." *Id.*, 10:34-39. Relatedly, Barsness teaches increasing data throughput by using processing elements "in parallel to process [sic] same type of data." *Id.*, 10:26-28. Barsness provides further description of how the management system duplicates or moves compute nodes to maintain expected performance levels. For example, if the "processing rate is above or below" a pre-determined level, the system takes "corrective actions" that include "spawning new compute nodes" and

"duplicating a compute node." 14:30-57; *see also id.*, FIG. 6, claim 2 ("adjusting the computing resources"), claim 5 ("cloning one of the processing elements" based on "processing state").

102. The management system 105 runs a node manager 340, as depicted in FIG. 3 of Barsness, "to configure the compute nodes," for example as part of "assigning or reassigning" processing elements "on different compute nodes." EX1004, 10:14-18; *id.*, 10:18-28 (describing the node manager "configuring each processing element and operator"); *see also id.*, 9:60-11:10 (describing management system CPU "configured to retrieve and execute programming instructions stored in the memory"). Regarding the virtual machine implementation of its system, Barsness describes the ability to "*spawn more virtual machines on the fly*" and thus provide "well known advantages" "including another layer of scalability." EX1004, 8:43-52 (emphasis added); *id.*, 7:45-48.

103. Applying these teachings in the context of the computing infrastructure shown in FIG. 1B, the management system is configured to provision a new compute node 130_x (*fourth computer system*) with processing elements PE1, PE2, and PE3 (*first transformation pipeline*) to address any "performance issues" that arise with compute node 130_1. EX1004, 10:30-34; *see also id.*, 10:34-39 (describing movement of "processing elements" among compute nodes "to improve the data processing performance of the stream application"). By processing the data in "parallel … to

increase data throughput" (EX1004, 10:26-28), this "duplicating a compute node" helps ensure an appropriate "processing rate" for the stream application (EX1004, 14:54-56, 14:30-39). The following annotated FIG. 1B shows duplicated / provisioned compute node 130_x (*fourth computer system*) configured with the processing elements of the first compute node 130_1 (*first transformation pipeline*), as taught by Barsness:

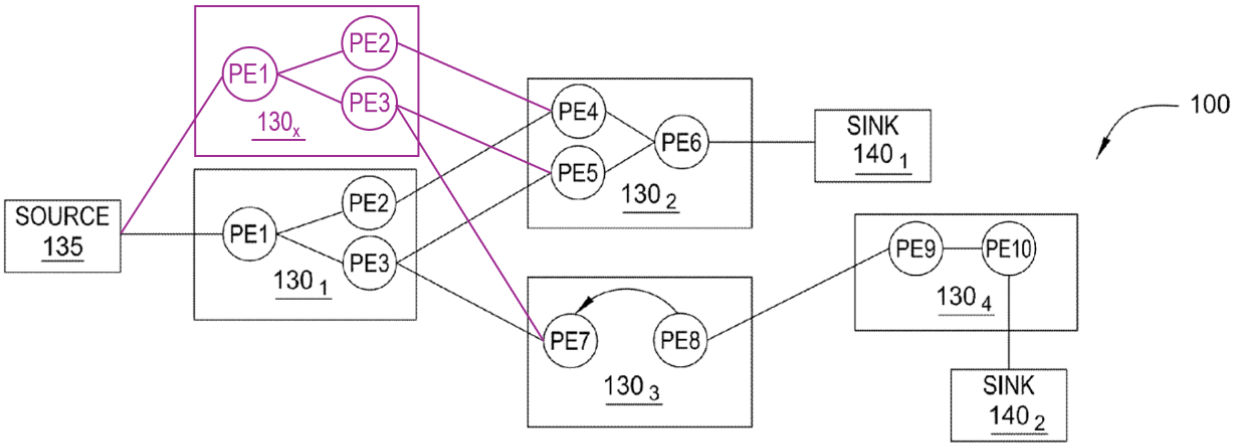


FIG. 1B

EX1004, FIG. 1B (purple annotations added).

104. Similar detection of performance issues for compute node 130_2 would lead the management system to duplicate that node, with duplicated / provisioned compute node 130_x (*a fourth computer system*) configured with the processing elements of the second compute node 130_2 (*the second transformation pipeline*). For example, if the monitoring system detects performance issues with compute node 130_2, the management system 105 provisions a new compute node 130_x (*fourth*

computer system) with processing elements PE4, PE5, and PE6 (*second transformation pipeline*) migrated / duplicated thereto "to improve the data processing performance of the stream application." EX1004, 10:34-39; *id.*, 10:28-34, 14:51-56 (describing automatically "duplicating a compute node"). Such a duplicated / provisioned compute node 130_x (*a fourth computer system*) configured with the processing elements of the second compute node 130_2 (*the second transformation pipeline*) is shown in this annotated FIG. 1B:

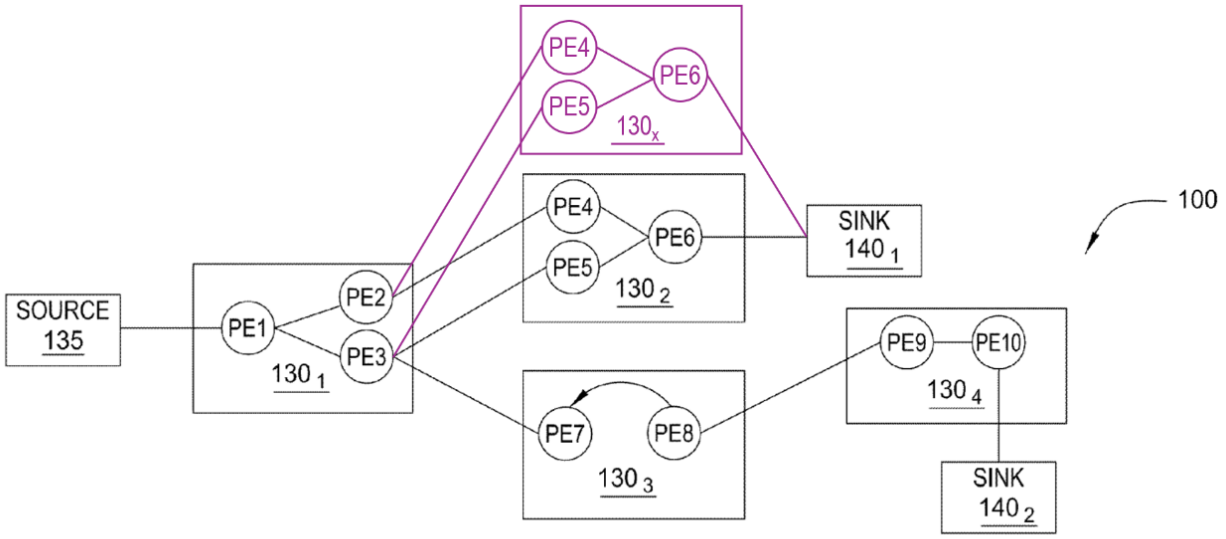


FIG. 1B

EX1004, FIG. 1B (purple annotations added).

105. Further, while Barsness discloses "spawning new compute nodes" and "duplicating a compute node" generally (EX1004, 14:51-56), a POSITA would have found it obvious to duplicate the compute nodes 130_1 and 130_2 in particular because Barsness discloses its management system being configured to duplicate any

compute node in the context of corrective measures for improving data processing throughput. EX1004, 10:26-39, 14:26-56. My analysis for claims 3 and 5 (below in Sections IX.C and IX.D) provides further examples of how the management system duplicates / spawns compute nodes to process data streams with the stream application.

106. Each of these newly spawned, duplicate compute nodes executes software instructions that apply at least one of the first transformation pipeline and the second transformation pipeline, as carried out by the respective processing elements. *See* paragraphs 74-79. Each duplicate compute node is of a second plurality of computer systems, for example a second plurality comprising the duplicate compute node along with the compute nodes of the first plurality of computer systems. As I noted above, the recited "second plurality of computer systems" in claim 1 is arbitrary because nothing says this second plurality is limited to only computer systems not found in the first plurality of computing systems or even that it is distinct from the first plurality. Even if it did, Barsness contemplates multiple additional compute nodes that are used as resources for provisioning nodes in order to meet system performance requirements, as explained above. These additional nodes also form a second plurality of computer systems, which includes duplicate compute nodes that perform the first transformation pipeline, the second transformation pipeline, or both.

107. Finally, the management system 105 is a *computer system of the first plurality of computer systems* as discussed above in paragraphs 50-52. The management system is a computer system including a CPU 305 that "retrieve[s] and *execute*[*s*] *programming instructions* stored in the memory 325 and storage 330," including a node manager 340. EX1004, 9:51-62 (emphasis added), 10:12-13. A detailed view of the management system, including the node manager as mentioned above, is shown in FIG. 3 of Barsness:

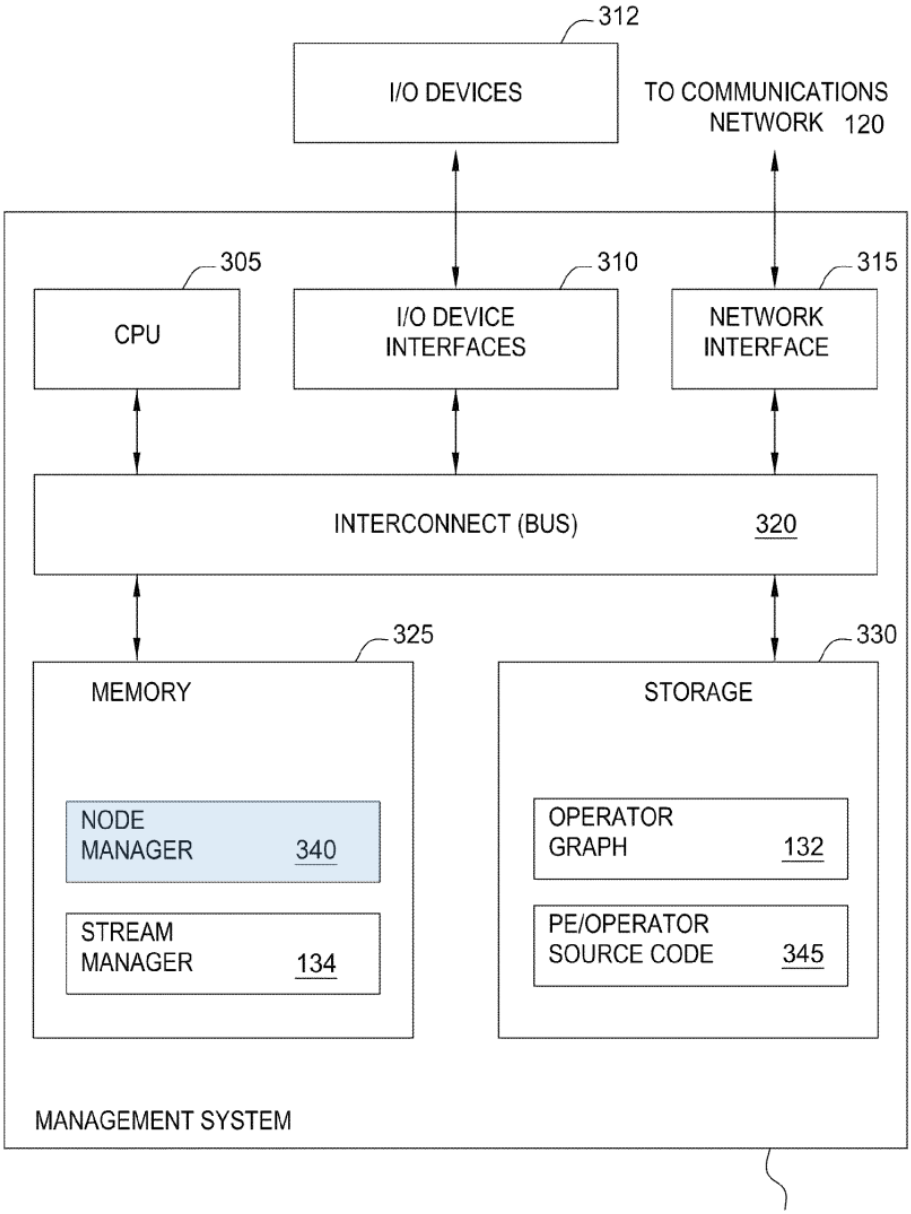


FIG. 3

EX1004, FIG. 3 (highlighting added). The management system executes the node manager "to configure the compute nodes 130 with respect to loading programming instructions and related configurations, assigning or reassigning processing elements

on different compute nodes 130, etc." EX1004, 10:14-18; *id.*, 10:18-28 (describing the node manager "configuring each processing element and operator").

B. Claim 2

> [2] The distributed computing cluster of claim 1, wherein at least one of the first transformation pipeline and the second transformation pipeline is non-linear.

108. Barsness renders this claim obvious. Barsness renders the distributed computing cluster of claim 1 obvious, as discussed above in Section IX.A (claim 1). Barsness discloses non-linear transformation pipelines. For example, as shown in FIG. 1B of Barsness, both the first transformation pipeline in the first compute node 130_1 and the second transformation pipeline in the second compute node 130_2 exhibit branching:

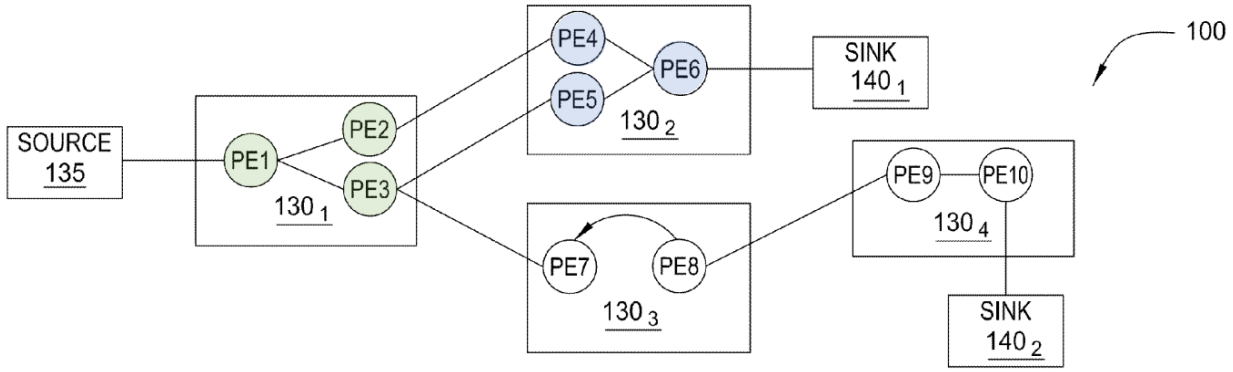


FIG. 1B

EX1004, FIG. 1B (highlighting added). Here, the operator graph shows efferent (fan-out) branching from PE1 to PE2 and PE3 and afferent (fan-in) branching from PE4 and PE5 to PE6. EX1004, 7:55-61, FIG. 1B. The operator graph also shows a

third transformation pipeline comprising the processing elements PE7 and PE8 in the third compute node 130_3, where "data tuples flowing to PE8 [] loop[] back to PE7," a cyclical flow of data. EX1004, 7:64-67, FIG. 1B. The fourth transformation pipeline comprising the processing elements PE9 and PE10 on the fourth compute node 130_4 is, by contrast, a linear transformation pipeline because the data flows linearly from PE9 to PE10.

109. Thus, at least one of the first transformation pipeline and the second transformation pipeline in an obvious implementation of Barsness is non-linear. Regarding non-linear transformation pipelines, the '424 patent indicates that pipelines configured with an "afferent branch," an "efferent branch," or a "cyclical" configuration are non-linear. EX1001, 10:4-7, FIGS. 13-15. As reflected in FIG. 13 of the '424 patent and its corresponding disclosure, an "afferent branch" configuration refers to output from each of two nodes flowing into the input of a third node, e.g., a fan-in arrangement. EX1001, FIG. 13, 21:42-60. Similarly, an "efferent branch" refers to a single node outputting to the inputs of a second and third node, e.g., a fan-out arrangement. EX1001, FIG. 14, 22:1-21. A "cyclical" configuration refers to the output of a node flowing back to a preceding node. EX1001, FIG. 15, 22:28-46.

C. Claim 3

> [3] The distributed computing cluster of claim 2, wherein the third computer system of the first plurality of computer systems is configured to:
>
> identify a fifth computer system of the first plurality of computer systems, wherein the fifth computer system is not configured to apply the first transformation pipeline to any stream of input data,
>
> and cause the fifth computer system to process a second stream of input data received from a second input feed substantially in real time by executing software instructions that apply the first transformation pipeline to the second stream of input data.

110. Barsness renders claim 3 obvious. Barsness renders obvious the distributed computing cluster of claim 2, as discussed above in Section IX.B. Further, Barsness discloses a management system (*the third computer system of the first plurality of computer systems*) that identifies additional compute nodes and provisions those compute nodes to, for example, duplicate existing compute nodes and "to improve the data processing performance of the stream application." EX1004, 10:34-39; *id.*, 10:30-34; *id.*, 8:45-48, FIG. 6, 14:30-56; *supra* paragraphs 100-107 (element 1.7). As explained in the following paragraphs, the Barsness teachings on node provisioning, scalability, and multiple input sources render obvious a management system configured to satisfy claim 3.

111. Barsness also teaches multiple input data streams, including a second stream of input data. For example, Barsness describes its operator graph representing "a stream application beginning from *one or more source processing elements*

(PEs)…" and being "configurable to *add or remove one or more PEs* based on the data processing needs and *scalability*." EX1004, 6:49-55 (emphasis added); *id.*, 2:52 (discussing "data sources"), 10:36-28 (multiple PEs "used in parallel to process same type of data to increase data throughput").

112. Regarding scalability, Barsness teaches addressing increased "data processing needs" (which arise with an additional input stream) by provisioning additional compute nodes. EX1004, 6:49-55; *id.*, 3:61-63 ("clone a given PEs" if "running too slowly to process data tuples through the stream"); *id.*, 6:49-55. Barsness teaches achieving scalability "by distributing an application across nodes … as well as replicating processing elements on multiple nodes and load balancing among them." EX1004, 3:19-22; *id.*, 6:16-42 (cloud computing infrastructure enables the "provision of scalable computing resources as a service over a network"). Similarly, Barsness teaches addressing processing needs by having the management system "spawn more process nodes … to process certain type of data" (EX1004, 10:28-31), and it expressly lists "spawning new compute nodes, duplicating a compute node, [and] spawning new processing elements" as example corrective measures (EX1004, 14:55-56). Collectively, these and other passages teach configuring the management system such that, when an additional input source stream is introduced, it duplicates/spawns existing compute nodes in order to scale and handle the additional stream and associated processing demand.

113. A POSITA would also have reasonably expected success with such an implementation because Barsness explains how to do so, as compute nodes may be "a physical host or a virtual machine" (EX1004, 8:37-39), virtualized deployments may "spawn more virtual machines on the fly" (EX1004, 8:47-48), and compute nodes may be provisioned by loading "programming instructions and configurations to be executed by the compute nodes" (EX1004, 7:13-15); *see also* EX1004, 10:12-42 (describing how system configures and sets up "processing elements and operators in the compute nodes 130"). Barsness further explains, in the context of its FIG. 1B example, that the management system "stream manager" is configured to "control the stream application by inserting, removing, or otherwise modifying the processing elements (or what data-tuples flow to the processing elements) running on the compute nodes $130_{1\text{-}4}$." EX1004, 7:33-37; *see also id.*, 7:29-33.

114. Barsness therefore expressly teaches the operator graph having multiple input streams and scaling the operator graph as needed to handle incoming data. *E.g.*, EX1004, 6:16-42; *id.*, 7:13-15, 7:29-37, 8:37-39, 8:47-48, 10:12-42. These contemplated additional input source streams of the same data type (e.g., due to increased volume or data coming from another location), predictably increase incoming data volume and/or the number of concurrent inputs the system must process. POSITAs would expect scenarios where the stream application needs to process multiple input streams, and thus scales accordingly, especially when using

a "cloud computing infrastructure," as taught by Barsness. EX1004, 6:16-42. For example, Heinze, an Association for Computing Machinery tutorial on scaling mechanisms in cloud-based stream processing systems, is a background reference reflecting the state of the art, and explains that "[c]loud-based data streaming engines, in particular, are designed to dynamically scale to hundreds of computing nodes and automatically cope with varying workloads." EX1008, 240-241. Heinze further discusses example data streams that are handled by data streaming engines, such as social network data streams, that "require systems to elastically scale and to support diverse workloads" because there may be "significant spikes" in data volume and data streams, including additional streams originating from around the world. EX1008, 238-241; *see also* EX1007, [0243] (adding nodes to handle "additional data streams"); *see below* Section XI.

115. Thus, Barsness contemplates multiple input streams and input nodes, and teaches scaling its streaming application to handle processing needs associated with additional input streams. Given these teachings, a POSITA would have been motivated with a reasonable expectation of success to configure a Barsness management system to, upon addition of a new input data stream: provision and configure an available compute node with processing elements for the first transformation pipeline (*identify a fifth computer systems of the first plurality of computer systems, wherein the fifth computer system is not configured to apply the*

first transformation pipeline to any stream of input data), and assign that duplicated provisioned node to process the second data stream with the first transformation pipeline (*cause the fifth computer system to process a second stream of input data received from a second input feed substantially in real time by executing software instructions that apply the first transformation pipeline to the second stream of input data*). *E.g.*, EX1004, 7:27-37, 10:12-64, 11:30-34, 14:51-56.

116. Barsness discloses *identifying a fifth computer system of the first plurality of computer systems* because the management system (*the third computer system*) manages the compute nodes and thus knows what compute nodes are available and not currently executing a transformation pipeline. EX1004, 10:12-64. To the extent that Barsness does not expressly disclose the management system identifying a fifth computer system, a POSITA would have been motivated with a reasonable expectation of success to implement the management system to do so in order to "manage the processing loads of the compute nodes 130," as taught by Barsness. EX1004, 7:27-33. For example, the management system is responsible for "assigning or reassigning processing elements on different compute nodes." EX1004, 11:30-34; *id.*, 14:51-56 (describing "automatic duplication of an operator" and automatically "spawning new processing elements"). Specifically, the management system is "configured for moving processing elements (PEs) from one compute node 130 to another, e.g., to manage the processing loads of the compute

nodes 130 in the computing infrastructure 100." EX1004, 7:29-33. The management system may also "control the stream application by inserting, removing, or otherwise modifying the processing elements (or what data-tuples flow to the processing elements) running on the compute nodes $130_{1\text{-}4}$." EX1004, 7:33-37. I note that Barsness teaches the monitoring system performing a number of automatic corrective actions (EX1004, 14:51-56) but also explains that the monitoring system accomplishes those actions (e.g., spawning new compute nodes, duplicating a compute node) by communicating with the management system. *E.g.*, EX1004, 14:30-35; 12:64-67 (the monitoring system interacts with the management system "to determine if rearranging some processing elements on computer nodes 130 may remedy the issue"). Given that the management system handles management and configuration of the compute nodes, the actions taken to dynamically adjust / scale the processing is performed by the management system, rather than the monitoring system.

117. The POSITA would have been motivated to implement the computing infrastructure in this way to beneficially provide system scaling, as needed (e.g., by duplicating the stream application by duplicating the compute nodes $130_{1\text{-}4}$ executing the stream application), for an additional data stream because this would handle data processing needs, as taught by Barsness. The POSITA would further be motivated to simplify state management of the data stream, because duplicating the

stream application allows the new data stream to be processed in real-time without imposing additional complexity and processing load on the compute nodes $130_{1\text{-}4}$. The fact that a POSITA would have easily and predictably implemented such a system is confirmed by the absence of any description in the ’424 patent of any computer systems causing another computer system to apply an existing transformation pipeline to a second stream of input data. Further, a POSITA would have reasonably expected success with such an implementation because Barsness explains how to do so, with its teachings on provisioning compute nodes. *E.g.*, EX1004, 7:13-15, 7:29-37, 8:37-39, 8:47-48, 10:12-42.

118. The following annotated Barsness FIG. 1B shows an example of the management system scaling compute nodes to handle a new input stream, with a provisioned compute node 130_x (*fifth computer system*) configured to apply the processing elements PE1, PE2, and PE3 (*first transformation pipeline*) to a second source of data (*a second stream of input data received from a second input feed*):

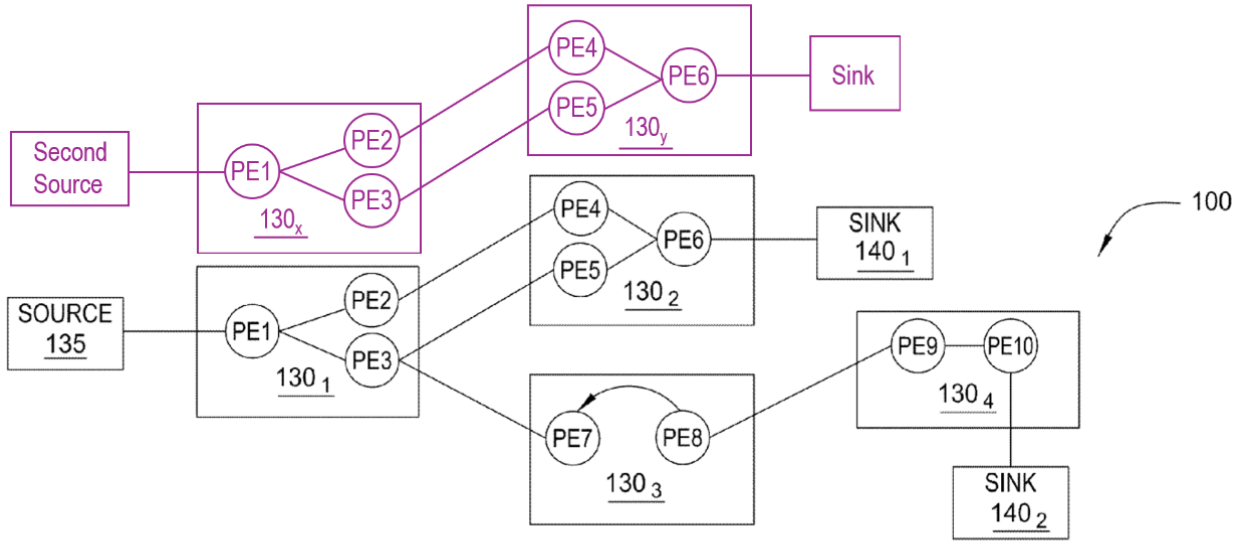


FIG. 1B

EX1004, FIG. 1B (purple annotations added). Though not depicted, a POSITA would have been motivated with a reasonable expectation of success to configure the management system to scale the entire streaming application, i.e., duplicate all compute nodes, to handle the increased processing needs of the second source. *E.g.*, *id.*, 7:29-37. This would result in duplication of third and fourth compute nodes 130_3 and 130_4 with PEs 7-10. These newly spawned compute nodes are a second plurality of computer systems, including compute node 130_x (*fifth computer system*) and compute node 130_y. Any two or more of these newly spawned compute nodes are a second plurality of computer systems, including compute node 130_x (*fifth computer system*) and compute node 130_y.

D. Claim 5

> [5] The distributed computing cluster of claim 2, wherein the third computer system of the first plurality of computer systems is configured to:
>
> identify a fifth computer system of the first plurality of computer systems, wherein the fifth computer system is not configured to apply the second transformation pipeline to any pipeline output message,
>
> and cause the fifth computer system to process third pipeline output messages substantially in real time by executing software instructions that apply the second transformation pipeline to the third pipeline output messages.

119. Barsness renders this claim obvious. Barsness renders obvious the distributed computing cluster of claim 2, as discussed above in Section IX.B. Further, Barsness discloses a management system (*the third computer system of the first plurality of computer systems*) that manages the creation of new compute nodes and the moving / replication of processing elements among compute nodes "to improve the data processing performance of the stream application." EX1004, 10:34-39; *id.*, 10:30-34; *id.*, 8:45-48; *supra* Sections IX.A.7 (element 1.7) and IX.C (claim 3).

120. As discussed above in Section IX.C (claim 3), a POSITA would have found it obvious to implement the management system of Barsness to cause a compute node, such as the compute node 130_x, to apply the first transformation pipeline to a second stream of input data. In that obvious implementation of Barsness, the output of this duplicated compute node 130_x applying the first

transformation pipeline comprises *third pipeline output messages*. Further, for the same reasons discussed above for claim 3, a POSITA would have been motivated to implement the management system 105 to cause the compute node 130_y (*fifth computer system*) to apply the processing elements PE4, PE5, and PE6 (*second transformation pipeline*) to the output data from PE2 and PE3 of the compute node 130_x (*process the third pipeline output messages substantially in real time by executing software instructions that apply the second transformation pipeline to the third pipeline output messages*) in order to process data from the second source with the duplicated stream application, as shown in this annotated FIG. 1B:

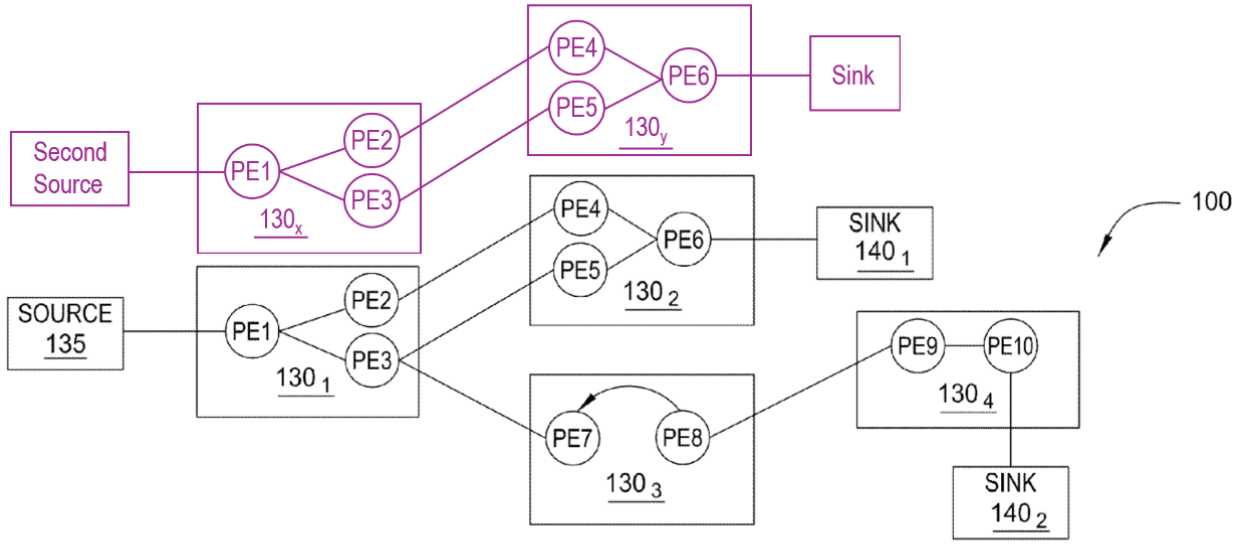


FIG. 1B

EX1004, FIG. 1B (purple annotations added); *id.*, 3:19-22 (describing achieving "[s]calability" of the stream application by "replicating processing elements on multiple nodes and load balancing among them"). In the alternative example where

the third compute node 130_3 hosts the second transformation pipeline, the compute node 130_y may duplicate the processing elements PE7 and PE8 and thus duplicate the second transformation pipeline. Further, while only the processing elements PE1 through PE6 are depicted in this example, a POSITA would have found it obvious to duplicate the transformation pipelines of the compute nodes 130_3 and 130_4 as well to complete the stream application as configured in the operator graph.

121. Barsness teaches, or at least renders obvious, configuring its management system (*the third computer system*) to "identify" a *fifth computer system of the first plurality of computer systems* for the same reasons explained for claim 3. *Supra* Section IX.C.

E. <u>Claim 7</u>

> [7] The distributed computing cluster of claim 1, wherein the first computer system is further configured to transmit the first pipeline output messages to the fourth computer system of the second plurality of computer systems,
>
> wherein the first computer system is distinct from the fourth computer system,
>
> and wherein the second computer system is distinct from the fourth computer system.

122. Barsness renders this claim obvious. Barsness renders claim 1 obvious, as discussed above in Section IX.A (claim 1). Further, as discussed above, Barsness discloses "automatic duplication of an operator, spawning new compute nodes, duplicating a compute node, spawning new processing elements, etc." to

automatically modify and manage the stream application. EX1004, 14:40-56. This includes "adding more processing elements to process the same type of data as being processed" to improve throughput of a given processing element / transformation pipeline, such as the second transformation pipeline. EX1004, 12:59-64.

123. Thus, POSITAs would have implemented Barsness such that, if the monitoring system detects performance issues with a compute node (such as compute node 130_2), the management system 105 provisions a new compute node 130_z (*fourth computer system*) with processing elements PE4, PE5, and PE6 (*second transformation pipeline*) to process the data "in parallel" and "improve the data processing performance of the stream application." EX1004, 10:26-39; *id.*, 14:51-56 (describing automatically "duplicating a compute node"). As illustrated using the annotated FIG. 1B operator graph below, with this "parallel" processing, the first compute node 130_1 (*first computer system*) transmits its output data (*first pipeline output messages*) to the duplicated / provisioned compute node 130_z (*a fourth computer system*) configured with the processing elements PE4-PE6 of the second compute node 130_2 (*the second transformation pipeline*):

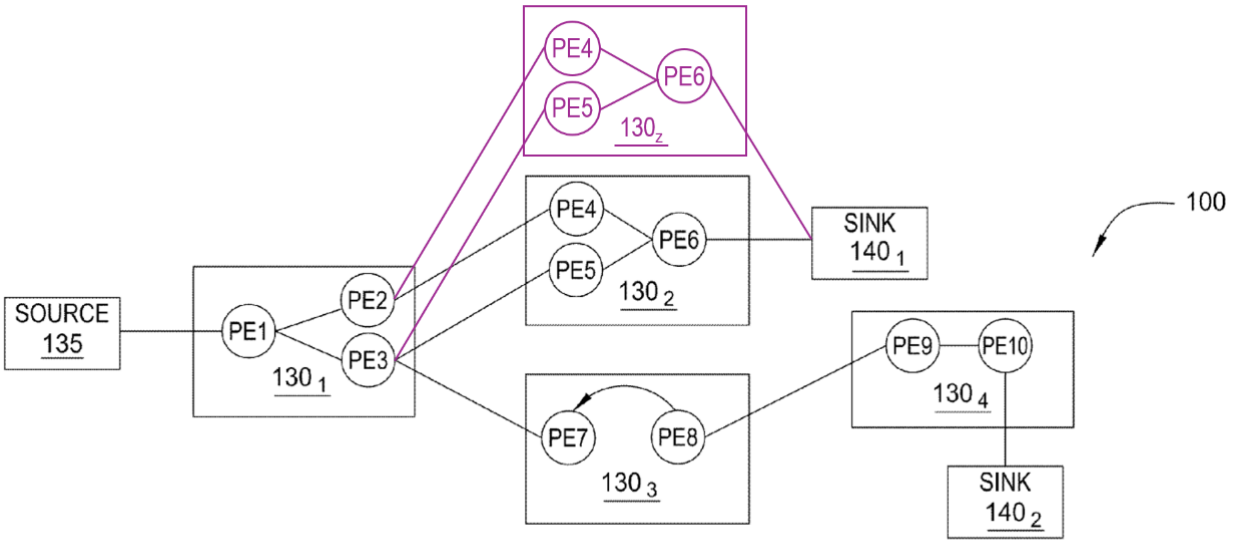


FIG. 1B

EX1004, FIG. 1B (purple annotations added). This parallel processing, with first compute node 1301 (*the first computer system*) transmitting output data from PE2 and PE3 (*first pipeline output messages*) to both duplicated processing elements PE4 and PE5 of the provisioned compute node 130_z (*fourth computer system*) in addition to the original processing elements PE4 and PE5 of the second compute node 130_2 (*second computer system*), beneficially "increase[s] data throughput" of the second transformation pipeline. EX1004, 10:26-28 ("More than one processing elements 235 and/or operators 240 may be used *in parallel to process same type of data to increase data throughput*.") (emphasis added); *id.*, 12:54-67 (describing adding additional operators / processing elements "to process the same type of data as being processed"). The duplicated / provisioned compute node 130_z thus processes the first pipeline output messages in parallel with the second compute node 130_2.

124. The Barsness teachings on "fault tolerance" also result in satisfaction of claim 7: if the second compute node 130_2 is failing, the management system duplicates and replaces the second compute node 130_2 with the compute node 130_z, to which the first compute node 130_1 transmits its output data messages instead of the failed compute node 130_2. EX1004, 8:45-52.

125. A POSITA thus would have been motivated to implement the Barsness system such that the first compute node (*first computer system*) is configured to transmit its output data (*the first pipeline output messages*) to the duplicated / spawned compute node (*fourth computer system of the second plurality of computer systems*) in order to improve performance of the stream application by supplementing (or replacing) the processing with the second transformation pipeline at the second compute node (*second computer system*) with the second transformation pipeline at the newly provisioned / duplicated compute node (*fourth computer system*).

126. The Barsness spawning/duplicating relies on "new compute nodes" (e.g., EX1004, 14:54-56) and thus a new node that is distinct from the existing nodes. EX1004, 14:51-56; *id.*, 8:43-44 (explaining that each virtual machine "runs as an *independent computer system*") (emphasis added). The new fourth compute node (*fourth computer system*) is therefore distinct from each of the first and second compute nodes (*first* and *second computer systems*). *E.g.*, EX1004, 6:16-42

(describing virtualization of the computing infrastructure), 8:37-52 (describing virtualization of compute nodes).

F. Claim 9

1. Element [9.1]

[9.1] A distributed computing cluster comprising:

127. Barsness satisfies the preamble because Barsness discloses a distributed computing cluster. *Supra* Section IX.A.1 (element 1.1).

2. Element [9.2]

[9.2] a first plurality of computer systems,

128. Barsness satisfies this element. *Supra* Section IX.A.2 (element 1.2). In the context of claim 9, Barsness discloses *a first plurality of computer systems* comprising at least a management system 105 and compute nodes 130_1 and 130_2. EX1004, 6:45-49. This first plurality of computer systems is shown in FIG. 1A of Barsness:

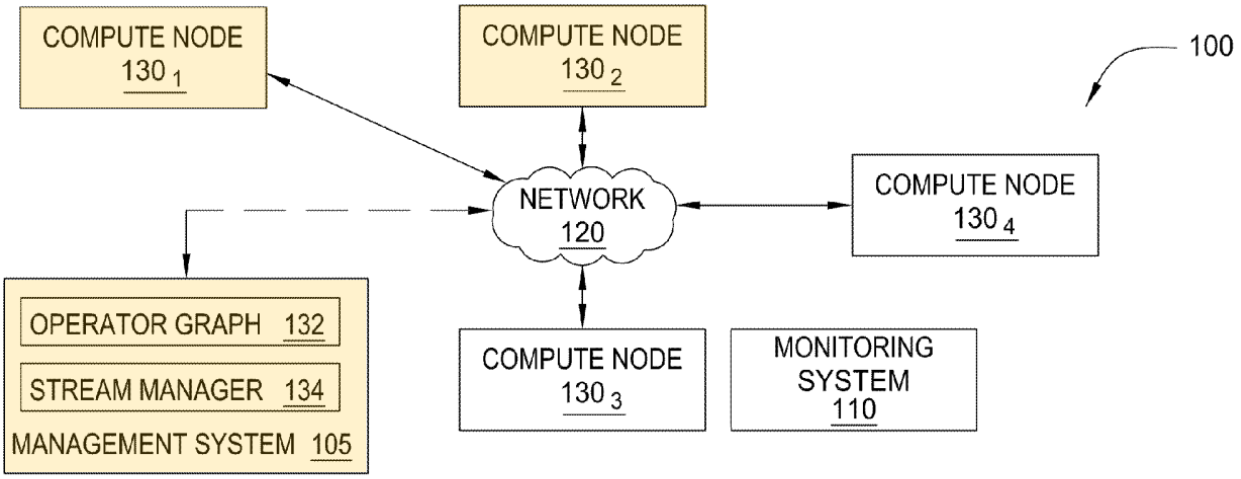


FIG. 1A

EX1004, FIG. 1A (highlighting added). Further, although not shown in FIG. 1A, the first plurality of computer systems further comprises compute nodes spawned by the management system 105 from the pool of computing resources. *Supra* Section IX.A.6 (element 1.6).

3. **Element [9.3]**

[9.3] wherein each respective computer system of the first plurality of computer systems comprises a respective memory that stores a respective first data that: describes at least a respective portion of a configuration of a first transformation pipeline, and identifies at least a second transformation pipeline,

129. Barsness satisfies this element because each of the management system 105 and compute nodes 130 (*each respective computer system of the first plurality of computer systems*) comprises a memory (*respective memory*) that stores respective programming instructions and application data (*respective first data*). *Supra* Section IX.A.2.a) (element 1.2.a). That data meets the additional recited claim

language here. *Supra* Section IX.A.2.b) (element 1.2.b). For example, the operator graph of Barsness describes *a configuration of a first transformation pipeline* because the operator graph represents a stream application that flows from source PEs to sink PEs, and shows the interconnection of PEs of a compute node applying a first transformation pipeline, as shown in FIG. 1B:

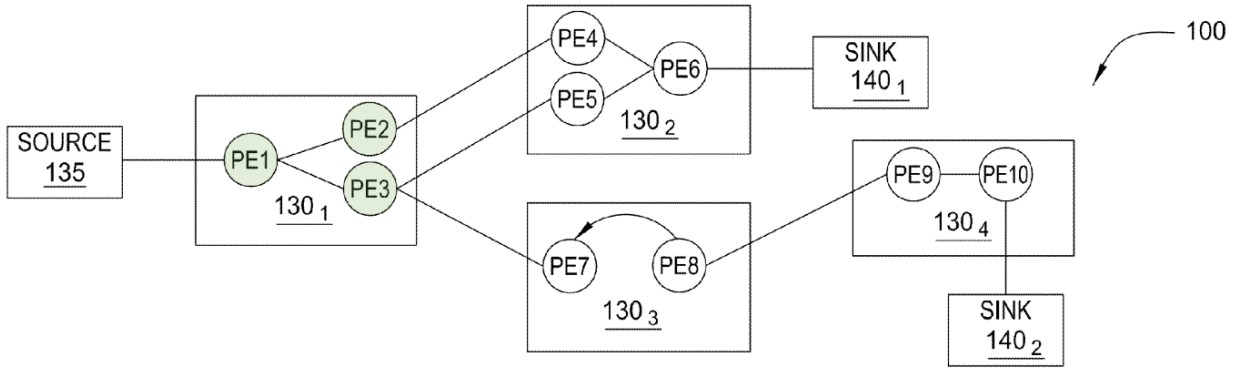


FIG. 1B

EX1004, FIG. 1B (highlighting added); *id.*, 6:50-56, 7:52-67 (describing the flow of data tuples through the nodes and processing elements of the example operator graph). In the context of the FIG. 1B example, the management system memory stores operator graph 132 that includes data that *describes at least a respective portion of a configuration of a first transformation pipeline* because it describes the configuration of PE1, PE2, and PE3 (*first transformation pipeline*). EX1004, 10:12-14, 10:48-56; *id.*, FIGS. 1B, 3.

130. Each compute node of the first plurality (e.g., 130_1 and 130_2) stores "programming instructions," "application data," buffered stream data, and other data

(*respective first data*) in its respective memory, including processing elements as designated by the operator graph. EX1004, 8:53-59, 8:66-9:33; FIG. 2. This "application data" (*respective first data*) specifies the connection to adjacent compute nodes (e.g., between compute node 130_1 and compute node 130_2), e.g., as the source and destination information of a TCP/IP socket, to transmit data between processing elements "in the stream application as described by the operator graph 132." EX1004, 9:4-9; EX1017, 1:22-2:26, FIG. 1; EX1016, 8:56-62. Each compute node thus stores data *describing at least a portion of a configuration of the transformation pipelines* that it receives data from and emits data to. The data stored on each compute node (e.g., compute node 130_1) also *describes at least a portion of a configuration of the transformation pipeline* on that particular compute node. *Supra* Section IX.A.2.b). For example, compute node 130_1 stores the processing elements PE1, PE2, and PE3 comprising the first transformation pipeline.

131. Thus, in the FIG. 1B example, at least first compute node 130_1 and the compute nodes it emits data to (130_2 and 130_3) store in memory data that *describes at least a portion of a configuration of the first transformation pipeline*. Similarly, if compute node 130_2 is viewed as the second transformation pipeline (*supra* Sections IX.A.3.d) and IX.A.4), then at least that compute node (130_2) and the compute node from which it receives messages (130_1) each has memory storing data that *identifies at least a second transformation pipeline* (the pipeline of compute

node 130_2). If compute node 130_3 is viewed as the second transformation pipeline, then it and 130_1 satisfy this element. *Supra* Sections IX.A.2.b) and IX.A.2.c).

4. <u>Element [9.4]</u>

> [9.4] wherein a first computer system of the first plurality of computer systems is configured to:
>
> receive a first stream of input data from a first input feed,
>
> process the first stream of input data substantially in real time by executing software instructions that apply the first transformation pipeline to the first stream of input data to generate first pipeline output messages,
>
> process the respective first data to determine an identification of the second transformation pipeline,
>
> and transmit the first pipeline output messages to a second computer system of the first plurality of computer systems in accordance with the determined identification,

132. Barsness satisfies this element. *Supra* Section IX.A.3 (element 1.3). The second "process …" and the "transmit …" aspects are largely similar to corresponding claim 1 language and are met for the same reasons. Namely, the Barsness FIG. 1B first compute node 130_1 (*first computer system*) is configured to process data stored in its memory (*the respective first data*) to identify the destination compute node PEs (*second transformation* pipeline) and to transmit its output message stream to that compute node (*transmit the first pipeline output messages to a second computer system of the first plurality of computer systems in accordance with the determined identification*). *Supra* Section IX.A.3.

5. Element [9.5]

[9.5] wherein the second computer system is configured to: receive the first pipeline output messages, and process the first pipeline output messages substantially in real time by executing software instructions that apply the second transformation pipeline to the first pipeline output messages,

133. Barsness satisfies these elements. *Supra* Sections IX.A.4 (element 1.4), IX.A.4.a) (element 1.4.a), IX.A.4.b) (element 1.4.b).

6. Element [9.6]

[9.6] wherein the first and second computer systems are distinct; and

134. Barsness satisfies this element. *Supra* Section IX.A.6 (element 1.5).

7. Element [9.7]

[9.7] a second plurality of computer systems;

135. Barsness satisfies this element. *Supra* Section IX.A.6 (element 1.6).

8. Element [9.8]

[9.8] wherein a third computer system of the first plurality of computer systems is configured to execute software instructions that cause the second plurality of computer systems to execute at least one of the first transformation pipeline and the second transformation pipeline.

136. Barsness satisfies this element. *Supra* Section IX.A.7 (element 1.7). The fourth compute node discussed above with regard to element 1.7 is part of the second plurality of computer systems.

G. Claim 10

> [10] The distributed computing cluster of claim 9, wherein one or more of the first transformation pipeline and the second transformation pipeline is non-linear.

137. Barsness renders this claim obvious. Barsness renders claim 9 obvious. *Supra* Section IX.F (claim 9). Further, Barsness discloses non-linear transformation pipelines. *Supra* Section IX.B (claim 2).

H. Claim 11

> [11] The distributed computing cluster of claim 10, wherein the third computer system of the first plurality of computer systems is configured to:
>
> identify a fourth computer system of the first plurality of computer systems, wherein the fourth computer system is not configured to apply the first transformation pipeline to any stream of input data,
>
> and cause the fourth computer system to process a second stream of input data received from a second input feed substantially in real time by executing software instructions that apply the first transformation pipeline to the second stream of input data.

138. Barsness renders this claim obvious for the same reasons discussed with regard to claim 3. *Supra* Sections IX.G (claim 10), IX.C (claim 3).

I. Claim 13

> [13] The distributed computing cluster of claim 9, wherein the first computer system is further configured to transmit the first pipeline output messages to a fourth computer system of the secondary plurality of computer systems,
>
> wherein the first computer system is distinct from the fourth computer system,

and wherein the second computer system is distinct from the fourth computer system.

139. Barsness satisfies these elements and thus renders this claim obvious for the same reasons discussed above with regard to claim 7. *Supra* Sections IX.F (claim 9), IX.E (claim 7).

X. GROUND 2: CLAIMS 1-3, 5, 7, 9-11, AND 13 ARE OBVIOUS OVER BARSNESS IN VIEW OF CHAKRADHAR

140. As explained above, Barsness alone renders claims 1-3, 5, 7, 9-11, and 13 obvious. *Supra* Section IX. These claims are further obvious in view of Chakradhar, which expressly teaches using route-maps and explains the benefits of doing so.

A. Chakradhar (EX1005)

141. Chakradhar is a U.S. patent application publication that published September 15, 2016, from an application filed March 14, 2016, and alleges priority to a provisional application filed on March 13, 2015 ("Chakradhar Provisional"). EX1005, Cover. Chakradhar is assigned to NEC Laboratories America, Inc.

142. Chakradhar is directed to the same field of distributed stream processing and provides an explicit mechanism for encoding and using topology information at the operator level. Chakradhar teaches that "[e]ach operator maintains a route-map that specifies the routes on which output messages should be sent after processing incoming messages," and provides examples in which route-map entries

identify downstream operators. EX1005, [0122]. Chakradhar further explains that route-maps encode "tuple-routing information and also the topology link structure." EX1005, [0054].

143. Chakradhar expressly ties the use of route-maps to concrete technical benefits. In particular, Chakradhar teaches that, "[b]y applying novel heuristics on the topology performance (feedback) metrics, … the controller determines efficient route-maps," and that these route-maps are "atomically injected into multiple operators, on-the-fly, using a light-weight, fault-tolerant protocol, for fast topology re-optimization." EX1005, [0054]. Chakradhar further explains that operators use these injected route-maps to control routing behavior during execution, enabling topology changes without interrupting stream processing. EX1005, [0054], [0122].

1. Chakradhar Is Supported By The Chakradhar Provisional

144. As I note below, the Chakradhar Provisional supports (1) at least one claim of Chakradhar, and (2) the portions of Chakradhar relied upon in this declaration.

a) The Chakradhar Provisional Supports Claim 1 Of Chakradhar

145. The chart below indicates disclosure in the Chakradhar Provisional that provided written description support for each element of claim 1 of Chakradhar.

Claim 1 of Chakradhar (EX1005)	Written Description Support in Chakradhar Provisional (EX1006)
A stream processing acceleration method employing network-aware routing, said method comprising the computer implemented steps of:	EX1006, 4/22: "Stream processing applications…" … "Stream processing engines…used to process continuous streams of data…" *Id.*, 5/22: "...a per-topology controller, which uses novel methods and protocols for dynamic, network-aware routing and on-the-fly topology modifications."… "These new route-maps are atomically injected into multiple operators, on-the-fly, using a light-weight, fault-tolerant protocol, for fast topology re-optimization." *Id.*, 13/22: "…the proposed MWD approach outperforms the random binary tree by a significant margin (more than two-fold speedup), for different model sizes." *Id.*, 14/22: "…we focus on heuristics and protocols for fast topology re-optimization, to maintain high stream-throughput at all times." *Id.*, 14/22: "…high-throughout stream processing…"
representing topology link structures using route-maps	*Id.*, 5/22: "(i) representing topology link structure using route-maps"
wherein said route-maps include tuple-routing information and topology structure encoded therein;	*Id.*, 5/22: "…the controller determines efficient route-maps, which encode tuple-routing information and also the topology link structure."
applying, on-the-fly, the route maps into multiple operators using a topology route-map update method.	*Id.*, 5/22: "…(iii) a light-weight, fault-tolerant protocol for atomic route-map update." … "These new route-maps are atomically injected into multiple operators, on-the-fly, using a light-weight, fault-tolerant protocol, for fast topology re-optimization."

Claim 1 of Chakradhar (EX1005)	Written Description Support in Chakradhar Provisional (EX1006)
	Id., 10/22: "…all operators start using the new route-maps for the same batch."

b) The Chakradhar Provisional Supports Relied-Upon Subject Matter Of Chakradhar

146. As demonstrated above, the Chakradhar Provisional discloses operators maintaining route-maps that include tuple-routing information and topology structure for directing output messages, i.e., the same technique disclosed in Chakradhar. In the discussion below, I also include citations to the Chakradhar Provisional supporting Chakradhar's relied-upon disclosure in the mapping of Barsness and Chakradhar to the challenged claims, which demonstrates that the Chakradhar Provisional supports all the relied upon-disclosure in Chakradhar.

B. Combining Barsness And Chakradhar

147. As discussed below, a POSITA would have been motivated, with a reasonable expectation of success, to combine the computing infrastructure for real-time stream processing applications taught by Barsness with the route-map teachings of Chakradhar to beneficially achieve "accelerated processing of stream processing engines" and "address[] bottlenecks in stream engines." EX1005, [0006]; EX1006, 4/22.

148. A POSITA implementing a system like that taught in Barsness would have been motivated to incorporate Chakradhar's route-map mechanism. Barsness

already requires operators to emit output to downstream PEs according to an operator graph, but does not prescribe a specific internal data structure for locally storing and updating routing information at each operator. EX1004, 9:4-9. Chakradhar addresses this known design need by explicitly teaching operator-local storage and processing of routing data structures representing at least portions of the distributed topology. EX1005, [0122]; EX1006, 9/22. A POSITA would have recognized that Chakradhar's route-maps provide a natural and efficient way to implement the routing behavior required by Barsness.

149. Moreover, the benefits identified by Chakradhar directly align with the goals and challenges described in Barsness. Chakradhar's ability to update routing behavior "on-the-fly" and "atomically" across operators supports fast reconfiguration, scalability, and fault tolerance, which Barsness identifies as important for stream-processing systems that must rebalance load or move processing elements between machines. EX1004, 3:56-60, 10:28-39; EX1005, [0054]; EX1006, 5/22. A POSITA would therefore have expected that incorporating Chakradhar's route-maps into Barsness would complement and improve the scalability, adaptability, and robustness of the computing system taught by Barsness. EX1004, 3:3-6 (describing "desired results with regards to scalability, efficiency, and real-time data processing"), 3:13-15 (describing "desired results where fault detections need to be fixed in real-time"), 3:19-22 (describing scalability).

150. Applying Chakradhar's route-maps to Barsness would have allowed each compute node to store respective data representing a portion of a distributed computational graph and to process that data locally to determine information identifying a downstream transformation pipeline and routing destination. Because Chakradhar's route-maps are expressly designed for distributed operator-based stream-processing systems, a POSITA would have had a reasonable expectation of success in making this modification. EX1005, [0043], [0056]; EX1006, 4/22, 6-7/22. The combination would have predictably reduced overhead for managing scalable stream applications and enabled efficient, consistent routing updates with minimal disruption to in-flight data streams. EX1005, [0054]-[0055]; EX1006, 5/22.

151. This obvious combination of Barsness and Chakradhar would include the features of the cloud-based cluster for distributed stream data processing applications taught by Barsness, as described *supra* Section VII, with the operators storing and maintaining route maps "which encode tuple-routing information and also the topology link structure" as taught by Chakradhar. EX1005, [0054]; *id.*, [0009]-[0010], [0053], [0122]; EX1006, 5/22, 9/22. So implemented, the Barsness-Chakradhar combination renders all challenged claims obvious.

C. <u>Claims 1-3, 5, 7, 9-11, And 13</u>

152. As discussed above in Section IX.A (claim 1), claim 1 is obvious in view of Barsness alone. Claim 1 is further obvious in view of the Barsness-

Chakradhar combination. Regarding a **distributed computing cluster** (element [1.1]), Chakradhar discloses "[s]tream processing engines" such as "Storm," which "is a distributed, real-time computation system" that "makes it possible to process unbounded streams of data." EX1005, [0042], [0056]; *id.*, FIG. 13 (depicting a "cloud-based stream processing system" with multiple incoming data streams), [0134]; EX1006, 4/22. Chakradhar explains that "[o]n a storm *cluster*, topologies are executed and process streams of tuples (i.e., data)." EX1005, [0056] (emphasis added); *see also id.*, [0046] ("The orchestration of cluster resources among these systems is handled by a global cluster scheduler such as Mesos."), [0069] ("route-maps are installed in a consistent manner on a running cluster"), [0098] ("nodes in a cluster"), [0148] (evaluation of techniques "conducted on a 30-node cluster"); EX1006, 4/22, 6-7/22, 11/22. These disclosures complement the distributed computing architecture of Barsness, and thus, in combination, the Barsness-Chakradhar combination includes a *distributed computing cluster*.

153. Regarding a **distributed computational graph** (element [1.2.b]), just as the operator graph of Barsness "represents a stream application" (EX1004, 6:49-53), Chakradhar explains that "stream engines code an application workflow as a *directed* acyclic graph (DAG) of operators, referred to as a topology." EX1005, [0043] (emphasis added); EX1006, 4/22. Chakradhar further discloses that stream data is "processed in a *pipelined* fashion as they traverse through the topology."

EX1005, [0043] (emphasis added); EX1006, 4/22. Chakradhar explains that "[t]he logic for *realtime* applications is packaged in a topology," and that "[e]ach topology is represented by a graph including spouts (which produce tuples) and bolts (which transform tuples)." EX1005, [0056] (emphasis added) ("As noted–with Storm–a topology is a graph whose nodes are operators (spouts and bolts) and edges are virtual connections among operators."); *id.*, [0106] (describing "a complete directed graph, where nodes denote machines hosting learner operators and edges represent potential overlay links"); EX1006, 5/22, 8/22.

154. Regarding **the respective first data [that] represents a respective portion of a distributed computational graph** (element [1.2.b]), Chakradhar discloses that "[e]very operator chooses a destination operator for its outgoing tuples based on route-maps." EX1005, [0069]; *id.*, [0070] (describing "tuple-routing" between operators); EX1006, 5-6/22, 14/22. Further, "[r]oute maps contain information on the type and proportion of traffic for each destination operator." EX1005, [0069]; EX1006, 6/22. Further, Chakradhar discloses that "[e]ach operator maintains a route-map that specifies the routes on which output messages should be sent after processing incoming messages." EX1005, [0122]; EX1006, 9/22. The route-maps taught by Chakradhar therefore "represent[] topology link structure." EX1005, [0053]; EX1006, 5/22. When route-maps are updated, they are "injected"

into the operators "for fast topology re-optimization." EX1005, [0010], [0054]; EX1006, 5/22, 14/22.

155. In combination, each compute node of Barsness-Chakradhar stores an operator (executable code corresponding to the task assigned to that processing element) as taught by Barsness (EX1004, 9:4-9) and a route-map specifying where "output messages should be sent after processing incoming messages" as taught by Chakradhar (EX1005, [0122]; EX1006, 9/22). Together, this data stored at the first compute node is *respective first data representing a respective portion of a distributed computational graph*.

156. Regarding **the first computer system processing the respective first data to determine information about the second transformation pipeline** (element [1.3.c]) and **transmitting the first pipeline output messages to a second computer system in accordance with the determined information** (element [1.3.d]), Chakradhar teaches that "[e]ach operator maintains a route-map that specifies the routes on which *output messages* should be sent after processing incoming messages." EX1005, [0122] (emphasis added); EX1006, 9/22. In combination, an operator and its route-map are stored in the memory of the first computer system of Barsness-Chakradhar, which enables the processing element / operator to emit / send its output messages to the appropriate downstream PE as described by the operator graph. *E.g.*, EX1004, FIG. 2 (showing operator in memory

of compute node). Specifically, such a route-map "encode[s] tuple-routing information and also the topology link structure," as taught by Chakradhar. EX1005, [0010], [0054]; *id*., [0043]; EX1006, 5/22, 9/22. In this way, "[e]very operator chooses a destination operator for its outgoing tuples based on route-maps." EX1005, [0069]; EX1006, 6/22. The first compute node, when executing the operator of a given processing element at the end of the first transformation pipeline (e.g., PEs 2 and 3 in example operator graph of Barsness FIG. 1B), thus processes the route-map (*the respective first data stored in the memory of the first computer system*) to determine the destination of its output message (*to determine information about the second transformation pipeline*). The operators of Barsness-Chakradhar use route-maps to identify the destination of their output messages, and thus the first compute node (*first computer system*) transmits the output of the first compute node (*the first pipeline output messages*) to the second computer system according to the destination(s) specified in the route-map(s) (*the determined information*).

157. Thus, claim 1 is obvious in view of the Barsness-Chakradhar combination. Claims 2, 3, 5, 7, 9-11, and 13 are obvious for the same reasons set forth above and in Ground 1. *Supra* Section IX.

XI. GROUND 3: CLAIMS 1-3, 5, 7, 9-11, AND 13 ARE OBVIOUS OVER BARSNESS IN VIEW OF SIRIPURAPU

158. As explained above, Barsness alone renders claims 1-3, 5, 7, 9-11, and 13 obvious. *Supra* Section IX. These claims are further obvious in view of Siripurapu, which expressly teaches adding computer nodes to handle processing additional real-time data streams and increased data flow.

A. Siripurapu (EX1007)

159. Siripurapu is a U.S. patent application publication that published February 27, 2014, and alleges priority to a provisional application filed November 18, 2010. EX1007, Cover.

160. Siripurapu is directed to the same field of processing real-time data streams with a "distributed, elastic architecture" and provides an explicit mechanism for handling "additional data streams" and "increased data stream flows." EX1007, [0060], [0453]; *id.*, [0359]-[0363]. Specifically, Siripurapu teaches that the "distributed implementation provides scalability, wherein one or more additional worker nodes may be added … to accommodate increased computational requirements, additional data streams, increased data stream flows, and the like." *Id.*, [0453]. Siripurapu explains that "[a]t any point in time, there are usually several Streams: some of them external (e.g., Twitter, Facebook) and others that are

generated as intermediate values and outputs by the application." EX1007, [0178]; *id.*, [0439]-[0440] (describing various "[i]nput real-time data streams").

161. Further, "[t]he distributed implementation also provides fault-tolerance so that there is no single point of failure, wherein a task at a failed worker node may be assigned to a different worker node without affecting the entire distributed system." EX1007, [0454]. Siripurapu also discloses assigning a stream to another compute node if the computational load of a compute node is too high, for example due to "hot spots in the flow of incoming stream events." EX1007, [0470].

162. As depicted in Figure 36, multiple "real-time data streams" arrive at a stream bus and are distributed to worker nodes by a conductor that manages the real-time processing of stream data by "act[ing] as a task scheduling mechanism to provide instructions to designate worker nodes for performing … operations." EX1007, [0444], [0446]; *id.*, [0448]-[0449] (explaining how the conductor monitors "performance of the worker nodes" and "perform[s] load balancing by reassigning tasks to the worker nodes"). For example, "[t]ask scheduling may be performed by the conductor in a manner to balance the loads among the worker nodes, prevent 'hot spots' from arising at any worker node and to reduce the response time required by each worker node to generate an output." EX1007, [0481].

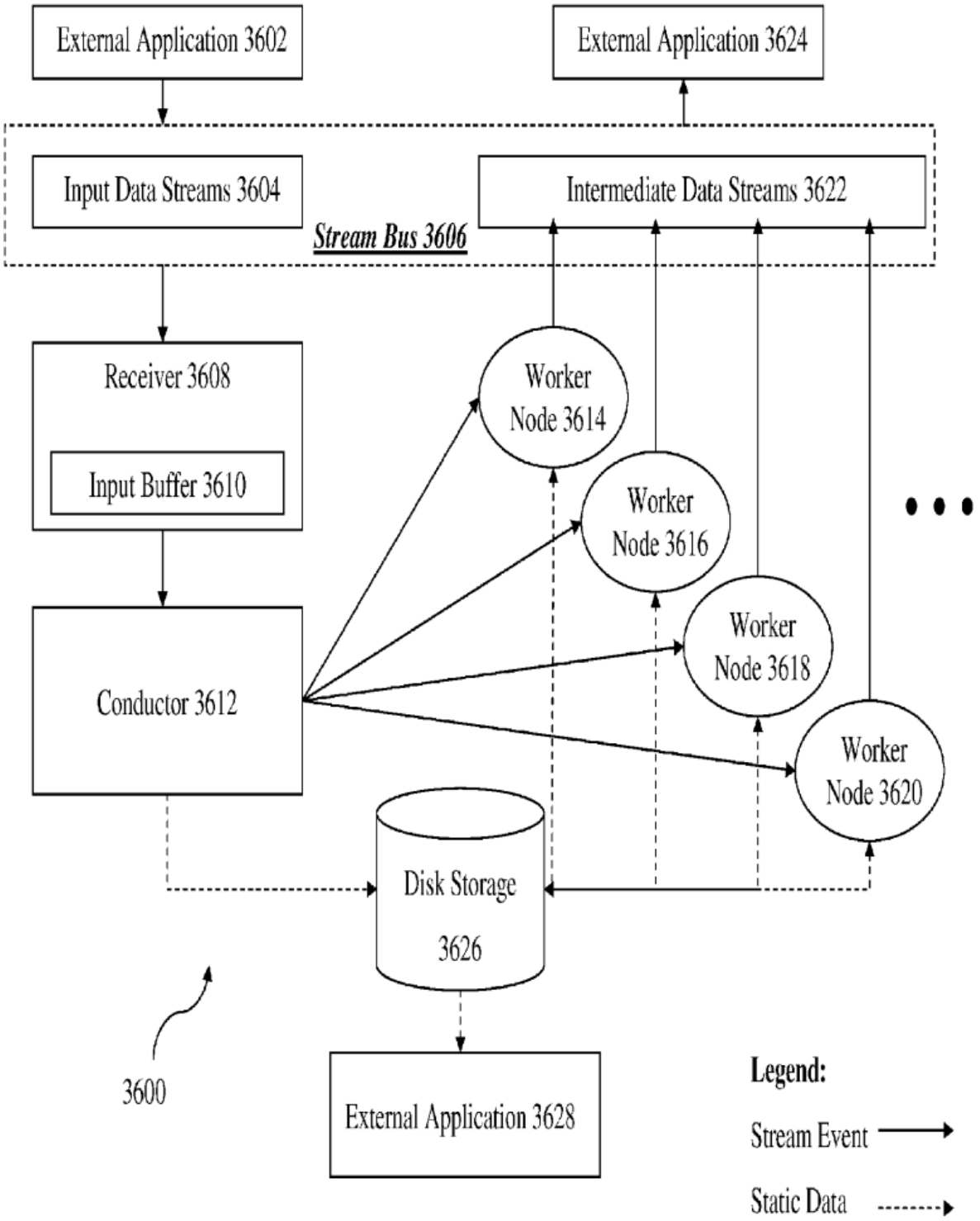


Figure 36

EX1007, FIG. 36.

B. Combining Barsness And Siripurapu

163. A POSITA would have been motivated, with a reasonable expectation of success, to combine the computing infrastructure for real-time stream processing applications taught by Barsness with the scalability teachings of Siripurapu to beneficially "accommodate increased computational requirements, additional data streams, increased data stream flows, and the like. EX1007, [0453].

164. Barsness is expressly concerned with scaling a stream-processing application across multiple compute nodes and generally describes the flexibility and adaptability of the distributed implementation to address data processing requirements. EX1004, 3:19-22 ("Scalability is achieved by distributing an application across nodes …"), 6:53-55 (PEs can be added or removed from operator graph "based on the data processing needs and scalability"). Siripurapu addresses the same practical scaling problem in a stream processing context, and provides a complementary, implementation-level rationale for scaling capacity when workload grows, where "one or more additional worker nodes" are dynamically added "to accommodate increased computational requirements, additional data streams, increased data stream flows, and the like." EX1007, [0453]. A POSITA reading Barsness and Siripurapu together would understand Siripurapu's additional worker nodes for additional data streams as directly aligning with the scalable infrastructure of Barsness that distributes execution of operators across compute nodes and can

add/remove processing elements for "data processing needs and scalability." EX1004, 6:53-55.

165. The scalability teachings of Siripurapu thus reinforce the teachings in Barsness regarding "replicating processing elements on multiple nodes and load balancing among them" (EX1004, 3:19-22) and spawning / provisioning more compute nodes as needed (EX1004, 6:19-42, 10:26-39). Like the management system of Barsness, the conductor of Siripurapu "perform[s] load balancing by reassigning tasks to the worker nodes" (EX1007, [0448]) and "assign[s] stream processing operations to the new worker nodes" provisioned to "accommodate increased computational requirements, additional data streams, increased data stream flows, and the like" (EX1007, [0453]).

166. Further, Siripurapu describes how a "distributed computational system" for providing real-time stream processing handles receiving "one or more real-time data streams" and distributing them to worker nodes executing operators. EX1007, [0439]; *id.*, [0440]-[0460] (further describing the elements of the distributed computational system). A POSITA implementing Barsness thus would have been motivated to combine the teachings of Siripurapu for handling additional input data streams and increased data stream flows, to thus ensure the scalability of the stream application when handling "the sheer volume of content available and the speed with which new content is made available." EX1007, [0002]; *id.*, [0352] (explaining

"spikiness in the flow of stream events in the input data streams" where "a particular topic or link may suddenly become 'hot' or popular in an input data stream, resulting in a sudden surge …").

167. Further, as Barsness already discloses a scalable cloud-based computing infrastructure, where compute nodes can be dynamically provisioned to handle data processing needs, a POSITA would have reasonably expected to succeed combining the teachings of Siripurapu with Barsness. EX1004, 6:16-42; *id.*, 3:19-22.

168. This obvious combination of Barsness and Siripurapu includes the features of the cloud-based cluster for distributed stream data processing applications taught by Barsness, as described *supra* Section IX, with the stream application scaling via the addition of compute nodes to accommodate additional data streams as taught by Siripurapu. EX1007, [0453]. These additional compute nodes would include the additional compute nodes I depicted in the annotated FIB. 1B of Barsness above in Sections IX.A.7, IX.C, and IX.D. So implemented, the Barsness-Siripurapu combination renders all challenged claims obvious.

C. <u>Claims 1-3, 5, 7, 9-11, And 13</u>

169. As discussed above in Section IX.A (claim 1), claim 1 is obvious in view of Barsness alone. Claims 1, 2, 7, 9, 10, and 13 are further obvious in view of the Barsness-Siripurapu combination, as a POSITA would have found it obvious to

implement the management system of Barsness to provision additional compute node(s) to handle increased data volume / processing needs at a given compute node/transformation pipeline, as taught by Barsness and Siripurapu. EX1007, [0453];]; *id.*, [0352] (explaining that teachings "are able to react in a fast and efficient manner to surges in input data streams, and to balance computational loads relating to the 'hot spots' so that the data structures and nodes are still able to operate efficiently and generate output in short response times"); *supra* Sections IX.A.7 (element 1.7), IX.B (claim 2), IX.E (claim 7), IX.F (claim 9), IX.G (claim 10), IX.I (claim 13).

170. Claims 3, 5, and 11 are further obvious in view of Barsness-Siripurapu, which includes Siripurapu's express teachings on provisioning additional compute node(s) to handle processing of one or more additional input data streams from different sources. EX1007, [0453], [0352]; *supra* Sections IX.C (claim 3), IX.D (claim 5), IX.H (claim 11).

XII. GROUND 4: CLAIMS 1-3, 5, 7, 9-11, AND 13 ARE OBVIOUS OVER BARSNESS IN VIEW OF CHAKRADHAR AND SIRIPURAPU

171. As explained above, Barsness alone renders claims 1-3, 5, 7, 9-11, and 13 obvious. *Supra* Section IX. For the same reasons a POSITA would have combined the real-time stream processing system of Barsness with the teachings of Chakradhar and Siripurapu individually (*supra* Sections X, XI), a POSITA would

have applied those teachings together when implementing Barsness. Those teachings relate to different aspects, are fully compatible with and complementary to one another: Chakradhar's relate to storing graph information on nodes to ensure proper and efficient downstream transmission of transformed data, and Siripurapu provides details on scalability of the system to handle different workload needs. Thus, a POSITA would have been motivated with a reasonable expectation of success to implement both simultaneously within the Barsness system. The challenged claims are further obvious in view of a combination of Barsness, Chakradhar, and Siripurapu.

I, Mark Crovella, do hereby declare and state, that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Dated: March 16, 2026 Signature

APPENDIX A

MARK E. CROVELLA
Duan Family Fellow
Professor, Computer Science
Professor, Faculty of Computing and Data Sciences

Boston University
665 Commonwealth Ave.
Boston, MA 02215

Current as of January 13, 2026

EDUCATION

Ph.D. Computer Science, University of Rochester, October 1994.

M.S. University of Rochester, May 1991.

M.S. State University of New York at Buffalo, May 1990.

B.S. Cornell University, May 1982.

PROFESSIONAL EXPERIENCE

Professor Department of Computer Science, Boston University, 2006 to present. **Department Chair** 2013 – 2018; **Associate Professor** 2000 – 2006; **Assistant Professor** 1994 – 2000. On leave 2000 – 2001.

Professor Faculty of Computing & Data Sciences, Boston University, 2022 to present. **Duan Family Fellow** 2024 – present; **Chair of Academic Affairs** 2022 – present; **Founding Faculty Member**, 2020 – 2022; **Data Science Faculty Fellow** 2019 – 2022.

Affiliated Faculty Department of Electrical and Computer Engineering, Graduate Program in Bioinformatics, Center for Information Systems and Engineering, and Division of Systems Engineering, Boston University.

Chief Scientist Guavus, Inc., February 2012 to August 2014.

Visiting Faculty Laboratoire d'Informatique de Paris 6 (LIP6); 2003 – 2004 and 2018 – 2019; INRIA Paris 2018 – 2019; Laboratory of Information, Networking and Communication Sciences (LINCS) Paris 2018 – 2019.

Technical Director Network Appliance, Tewksbury, MA. November 2000 to April 2002.

Chief Architect Webmanage Technologies, Chelmsford, MA. May 2000 to acquisition by Network Appliance in November 2000.

Lead Research Scientist epicRealm, Inc., Richardson, TX. January to May 2001.

Senior Computer Scientist Calspan Corporation, Buffalo, NY. August 1984 to August 1994.

HONORS

Duan Family Faculty Fellow, 2024, Boston University.

Distinguished Alumni Award, 2017, University at Buffalo, Department of Computer Science and Engineering.

Innovator of the Year, 2014, Boston University.

IETF/IRTF Applied Networking Research Prize for "Routing State Distance: A Path-based Metric for Network Analysis," ACM Internet Measurement Conference (IMC), 2012.

Best Student Paper Award for "Intrusion as (Anti)social Communication: Characterization and Detection," ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD) 2012.

IEEE Fellow, 2011, "For contributions to the measurement and analysis of networks and distributed systems."

ACM Fellow, 2010, "For contributions to the measurement and analysis of networks and distributed systems."

ACM SIGMETRICS Test of Time Award for "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes," (with Azer Bestavros). One of the three inaugural Test of Time Awards given by SIGMETRICS in 2010. The award is given for a paper published between 1973 and 1999 that has had the greatest impact over the subsequent period.

CNRS Fellowship, Centre National de la Recherche Scientifique (Department Sciences et Technologies de l'Information et de la Communication), for 2003-2004 academic year as Visiting Associate Professor at Laboratoire d'Informatique de Paris 6 (LIP6).

NSF CAREER Award, 1995.

ARPA Fellowship in High Performance Computing, 1993-1994.

DARPA Parallel Processing Fellowship, 1992-1993.

Editorial Boards

Steering Committee, IEEE Transactions on Network Science and Engineering, 2016–2019.

Editor, Computer Communication Review, 2005 – 2008.

Editor, Computer Networks, 2004 – 2005.

Editor, IEEE/ACM Transactions on Networking, 2000 – 2006.

Editor, IEEE Transactions on Computers, 2000 – 2004.

Funding

"Mechanistic Interpretability of Large Language Models," Open Philanthropy, 2025 – 2026, $52,646.

"Large Scale Analysis of Configurations and Management Practices in the Domain Name System," NSF CISE/CNS NeTS Medium, 10/1/23 – 9/30/26, $525,000.

"Methods for Active Measurement of the Domain Name System," NSF CISE/CNS IMR, 10/1/23 - 9/30/26, $256,463.

Google Research Award, 10/1/20, $30,000.

"Traveling Back to the Future: Using Network Science to Unravel Evolutionary Conserved RNA Binding Proteins and Complexes Associated with Disease," BU Hariri Institute Research Incubation Award, 2/9/18, $35,280.

"Analytic Tools for Evolving Path-based Networks," NSF CISE/CNS NeTS Small, 8/15/16 – 7/31/19, $499,951.

"Structural Matrix Completion for Data Mining Applications," NSF CISE/III Small, 9/1/14 – 8/31/17, $499,702 with 1 other PI.

"Understanding Communication Strategies for Ad Hoc Networks," NSF CISE/CNS NeTS Small, 8/1/11 – 7/31/14, $390,853.

"Information and Software Assurance: A Coordinated Approach to Cyber-Situation Awareness Based on Traffic Anomaly Detection,", DoD ARO, 9/1/2011 - 8/31/2015, $800,000 with 3 other PIs.

"Securing the Open Softphone," NSF CISE/CNS TC Large, 7/1/10 – 6/30/15, $2,992,896, with 8 other PIs. REU Supplement, 4/13/11 – $73,125.

"New Directions in Network Dimensionality Reduction for Routing and Beyond," NSF CISE/CNS NeTS Small, 7/1/10 – 6/30/13, $450,000.

"Wide-Aperture Traffic Analysis for Internet Security," NSF CISE/CNS TC Medium, 9/1/09 – 8/31/13, $723,053, with 2 other PIs.

Sponsoring Advisor for Postdoctoral Fellow under the *Computing Innovations Fellowship Program,* NSF CIF, 9/1/09 – 8/31/11, $267,500.

"Passive Methods for Internet Topology Discovery," National Geospatial Intelligence Agency, 8/1/09 – 7/31/12, $360,000.

"Instrumentation and Measurement for GENI," NSF/BBN, 8/1/08 – 8/31/11, $80,932

"Whole-Network Anomaly Diagnosis," Intel, 1/1/05 – 12/31/07, $158,000.

"Coordinate Systems for the Internet," NSF CISE/ANI, 8/15/03 – 7/31/06, $222,035.

"Collaborative Research: Modular Strategies for Global internetwork Monitoring," with 6 other PIs, NSF CISE/CCR ITR, 9/1/03 – 8/31/08, $1,994,513.

"Research Support for IP Network Characterization," Sprint Advanced Technology Laboratories, 9/1/03 – 8/31/04, $29,592.

"Research Support for IP Network Characterization," with 3 other PIs, Sprint Advanced Technology Laboratories, 9/1/02 – 8/31/03, $80,000.

"A Control Theoretic Approach to the Design of Internet Traffic Managers," with 3 other PIs, NSF CISE/ANIR, 9/1/01 – 8/31/05, $719,994.

"Diagnosis and Control of Network Variability by Massively Accessed Servers," with 2 other Principal Investigators, NSF CISE/ANI, 9/1/00 – 8/31/05, $1,210,768.

"Wide Area Web Measurement," GTE Laboratories, 9/1/00 – 8/31/01, $20,000.

"Wide Area Web Measurement," Xerox PARC, 9/1/99 – 8/31/00, $15,000.

"COMMONWEALTH: Architecture and Protocols for Scalable WWW Service," with 2 other Principal Investigators, NSF CISE/CCR Experimental Software Systems, 7/15/97 – 6/30/00, $590,087.

"Research Support for Network Performance Characterization, Analysis, and Evaluation," Principal Investigator, Hewlett-Packard Company, 9/1/96 – 9/1/97: $69,404. Renewed for 9/1/97 – 9/1/98: $88,154. Renewed for 9/1/98 – 9/1/99: $29,873.

"Practical Performance Prediction for Parallel Programmers," Principal Investigator, NSF CISE/CCR CAREER, 9/1/95 – 5/31/98, $91,066

"Research Infrastructure for Parallel and Distributed Systems: Real-Time, Multimedia, and High-Performance," with 4 other Principal Investigators, NSF CISE Institional Infrastructure, $874,019

"Research Instrumentation for Real-Time, Multimedia and High-Performance Computing in Distributed Systems," with 4 other Principal Investigators, NSF CISE IGR, $142,744

"Graduate Assistantships in High Performance Computing," with 5 other Principal Investigators, US Dept. of Education, 9/1/95 – 8/31/98, $354,645

Professional Affiliations

Association for Computing Machinery (Fellow)
IEEE (Fellow)
ACM SIGCOMM and SIGMETRICS
IEEE Computer Society and Communication Society

Invited Conference Talks

Invited Speaker, *TMA Experts Summit,* Paris, June 18, 2019.

Invited Panelist, *The 3rd Future Network Development Conference,* Nanjing, China, May 22, 2019.

Keynote Speaker, *IEEE Workshop on Data Science for Networking (DS4N),* Boston, MA, December 14, 2017.

Invited Panelist, *CSE 50th Anniversary Celebration,* University at Buffalo, Department of Computer Science and Engineering, September 30, 2017.

Keynote Speaker, *DATANET 16: Workshop on Data-Driven Networking,* Arizona State University, May 13, 2016.

Invited Speaker, *Workshop on Mathematics of Data Analysis in Cybersecurity,* ICERM, Brown University, October 24, 2014.

Invited Speaker, *51st Annual Allerton Conference on Communication, Control, and Computing,* University of Illinois at Urbana-Champaign, October 3, 2013.

Invited Speaker, *Network Links: Connecting Social, Communication and Biological Network Analysis,* Institute for Mathematics and its Applications (IMA), Minneapolis, MN, February 29, 2012.

Invited Speaker, *Statistics of Networks Workshop,* Isaac Newton Institute for Mathematical Sciences, Cambridge, UK, June 24-25, 2010.

Keynote Speaker, *NetEval 2009 Workshop, The First Workshop on Performance Evaluation of Next-Generation Networks,* Boston, MA, April 26, 2009.

Keynote Speaker, *2006 Euro-NGI Workshop on Visions of Future Generation Networks (EuroView2006),* Würzburg, Germany, August 1, 2006.

Invited Speaker, *Maryland Hybrid Networks Center,* Advanced Networks Colloquium Series, University of Maryland, December 16, 2005.

Keynote Speaker, *Workshop on End-to-End Monitoring Techniques and Services (E2EMON),* Nice, France, May 15, 2005.

Invited Speaker, *Workshop on Internet Measurement, Modeling, and Analysis,* Statistical Research Center for Complex Systems (SRCCS), Seoul National University, Seoul, South Korea, January 11, 2005.

Invited Speaker, *Workshop on Measurement, Modeling and Analysis of the Internet,* Institute for Mathematics and its Applications (IMA), Minneapolis, MN, January 12, 2004.

Keynote Speaker, *First Workshop on Performance of Computational Systems and Communication,* part of the XXII Congress of the Brazilian Computer Society, Florianopolis, Brazil, July 18, 2002.

Invited Speaker, Institute for Pure and Applied Mathematics (IPAM) Program on Large Scale Communication Networks, Los Angeles, CA, March 18-22, 2002.

Invited Speaker, *DIMACS Workshop on Internet and WWW Measurement, Mapping and Modeling,* New Brunswick, NJ, February 13-15, 2002.

Invited Speaker, *Workshop on Job Scheduling Strategies for Parallel Processing,* Cambridge, MA, June 16, 2001.

Invited Speaker, *COST 257 Final Conference,* Würzburg, Germany.

Invited Plenary Speaker, *Performance Tools 2000:* The 11th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation, Schaumburg, Illinois, March 27 - 31, 2000.

Invited Speaker, *Institute for Mathematics and Its Applications Workshop: "Scaling Phenomena in Communication Networks,"* Minneapolis, MN, October 22-24, 1999.

Invited Speaker, Boston University Minority Engineers' Society, 2nd Annual Technical Exposition, Boston, MA, April 10, 1999.

Invited Speaker, Workshop on Intelligent Internet Infrastructure, IBM T.J. Watson Laboratory, Yorktown Heights, NY, March 22, 1999.

Invited Speaker, Workshop on Architectures, Performance, and Applications in Globally Distributed Web Services, Federal University of Minas Gerais (UFMG), Belo Horizonte, Brazil, March 11 - 12, 1998.

Invited Speaker, Long-Range Dependence and Self-Similar Network Traffic Session, Winter Simulation Conference, Atlanta, GA, Dec. 7-10, 1997

Invited Speaker, DIMACS/NSF Workshop on "End-to-End Network Modeling and Simulations," Princeton University, Princeton, NJ, October 23-25, 1997.

Invited Panelist, Hot Topic Session On Traffic Characterization, ITC-15, Washington DC, June 22-27, 1997

Invited Speaker, IFIP Working Group 7.3 Workshop, June 19-20, Orcas Island, WA.

Invited Panelist, "Engineering Implications of Long-Range Dependence in Teletraffic Data," MASCOTS '97, January 1997, Haifa, Israel

Invited Speaker, Applied Probability Session of Informs, the Operations Research Society, May 1997, San Diego, CA.

Invited Speaker, Joint Research Conference on Statistics in Quality, Industry and Technology, May 29-31 1996, NIST, Gaithersburg, MD. Sponsored by the *American Statistical Association*, the *Institute of Mathematical Statistics*, E. I. du Pont, and the *National Institute of Standards and Technology.*

Professional Activities

Chair, Scientific Advisory Board of the Max Plank Institute for Software Systems, 2024.

Scientific Advisory Board of the Max Plank Institute for Software Systems, 2016 – 2025.

Member, IETF/IRTF Applied Networking Research Prize Selection Committee, 2014 – 2016.

Executive Steering Committee of the Hariri Institute for Computing and Computational Science & Engineering, 2011 – 2014, 2024 – present.

Member of the External Review Visiting Committee for the Department of Computer Science at Amherst College, February 2013.

Organized Workshop: "GPU Programming with CUDA" at Boston University, July 24, 2012.

Member, SIGCOMM Rising Star Award Selection Committee, 2011.

Chair, SIGCOMM Test of Time Award Selection Committee, 2011.

Team Mentor for Conrad Awards, Cyber Security Challenge 2011.

External Judge for FCC's Open Internet Apps Challenge, 2011.

General Co-Chair, CoNEXT 2011.

Internet2 perfSONAR Workshop, Executive Committee, 2010.

Chair, ACM SIGCOMM 2007–2009.

Program Co-Chair, IFIP Networking 2010.

Internet2 Network Research Strategic Planning Team, 2009-present.

Internet2 Network Research Review Committee, `http://www.internet2.edu/networkresearch/nrrc.html`, 2009-present.

Advisory Board of CRAWDAD (Community Resource for Archiving Wireless Data At Dartmouth).

Steering Committee, Passive and Active Measurement Workshop (PAM), 2004-2006.

General Chair, Passive and Active Measurement Workshop (PAM) 2005.

Steering Committee, Internet Measurement Conference, 2003-2007.

ACM SIGCOMM Executive Committee, Award Committee Chair, 2003-2006. b

Program Comittee Chair, Internet Measurement Conference 2003.

Co-organizer (with C. Lindemann and M. Reiser), Workshop on Internet Performance Modeling, Dagstuhl Castle, Germany, September 26-29, 1999.

Co-organizer (with five others), BU/NSF Workshop on Internet Measurement, Instrumentation, and Characterization; Boston University, August 30, 1999.

Tutorials Chair, SIGMETRICS '99, Atlanta GA.

Program Committees:

- SIGCOMM: 2002, 2006, 2008, 2012, 2017
- SIGMETRICS: 1999, 2000, 2002, 2007, 2009, 2010, 2011, 2012, 2014, 2015, 2016
- CoNEXT: 2005, 2007, 2010, 2011

- Internet Measurement Conference: 2002, 2004, 2007, 2010, 2012, 2015, 2017, 2018, 2020, 2022
- ICDM: 2018
- The Web Conference: 2000, 2001, 2002, 2019, 2022 (Senior PC Member)
- KDD: 2017, 2023
- Intelligent Systems for Molecular Biology (ISMB): 2015, 2017
- First IMC Workshop on Internet Visualization: 2012
- Performance: 2011
- Passive and Active Measurement Workshop (PAM): 2009, 2010
- Infocom: 2009 (TPC Area Chair), 2010 (TPC Area Chair)
- IFIP Networking: 2009
- NetEval: 2009
- HotMetrics: 2008
- E2EMON: 2007
- First Workshop on Experimental Computer Science: 2007
- Workshop on Mining Network Data (MineNet-06): 2006
- Workshop on Large Scale Network Inference: 2005
- 6th Global Internet Symposium: 2001
- 6th International Workshop on Web Caching and Content Distribution: 2001
- Workshop on Caching, Coherence and Consistency: 2001
- Workshop on Network-Related Data Management: 2001
- ITC Specialist Seminar on IP Traffic Management and Modeling: 2000
- SPIE Performance and Control of Network Systems Conference: 1998, 1999
- 18th International Conference on Distributed Computing Systems (ICDCS-98): 1998
- Third IEEE Workshop on Services in Distributed and Networked Environments (SDNE '96): 1996

Course Texts and Materials

"Linear Algebra, Geometry, and Computation," Electronic Text, one-semester course on Linear Algebra from a Computer Science standpoint. 2015-2025. Available at `http://mcrovella.github.io/CS132-Geometric-Algorithms`. Includes contributions from Wayne Snyder and students.

"Computational Tools for Data Science," Electronic Text, one-semester course on applied methods in machine learning using Python. 2021-2022. Available at `http://mcrovella.github.io/CS506-Computational-Tools-for-Data-Science`. Includes contributions from Evimaria Terzi and George Kollios.

"Foundations of Data Science III," Electronic Text, one-semester course on probability and statistics for data science, 2023. Available at `https://mcrovella.github.io/DS122-Foundations-of-Data-Science-III/`. Co-developed with Pawel Przytycki and with contributions by Lisa Wobbes.

Tutorials and Short Courses

"Data Mining for Measurement Data," presented at the 5th PhD School on Traffic Monitoring and Analysis, associated with the 7th International Workshop on Traffic Monitoring and Analysis, Barcelona, April 21-22, 2015. Available at `https://github.com/mcrovella/mining-low-dim-network-data`.

"Traffic Modeling: Methods and Results for Single Links and Whole Networks," presented at SIGCOMM 2004, Portland OR.

"Traffic Modeling," short course presented at Aalborg University, February 2, 2004.

"Performance Characteristics of World Wide Web Information Systems," presented at SIGMETRICS '97, Seattle WA, June 1997; SIGMETRICS '98, Madison WI, June 1998; and Performance Tools '98, Palma de Mallorca, Spain, September 1998.

CONSULTING

Member of the Technical Advisory Board, Guavus, Inc. 2006 to 2017.

Consultant, Fujitsu Laboratories of America, Sunnyvale, CA. 2002 to 2004.

Consultant and Expert Witness, in litigation cases involving software patents, software trade secrets, and intellectual property. Ongoing.

Member of the Technical Advisory Board, epicRealm, Inc., Richardson, TX. March 2000 to May 2001.

Consultant and Member of the Technical Advisory Board, WebManage Technologies, Inc., Chelmsford MA. July 1999 to November 2000.

Founder and Principal Technical Officer, Commonwealth Network Technologies, Inc., Boston MA. Incorporated April 27, 1998; acquired by WebManage Technologies, October 12, 1999.

Consultant and Member of the Technical Advisory Board, VitalSigns Software Inc., Santa Clara, CA; acquired by International Network Services, Inc.; now part of Lucent. November 1996 to October 1999.

PATENTS

Methods for Storing and Reading Digital Data on a Set of DNA Strands, US Patent # 10,027,347, issued July 17, 2018.

Method and Apparatus for Whole-Network Anomaly Diagnosis and Method to Detect and Classify Network Anomalies Using Traffic Feature Distributions, US Patent # 8,869,276, issued October 21, 2014.

Compressed Set Representation for Sets as Measures in OLAP Cubes, US Patent # 8,533,167, issued September 10, 2013.

Method and Apparatus for Dynamic Resource Discovery and Information Distribution in a Data Network, US Patent # 7,747,741, issued June 29, 2010.

System for Creating and Distributing Prioritized List of Computer Nodes Selected as Participants in a Distribution Job, US Patent # 7,346,682, issued March 18, 2008.

Method and Apparatus for Election of Group Leaders in a Distributed Network, US Patent # 6,993,587, issued January 31, 2006.

Method and Apparatus for Scalable Distribution of Information in a Distributed Network, US Patent # 6,748,447, issued June 8, 2004.

Method and Apparatus for Reliable and Scalable Distribution of Data Files in Distributed Networks, US Patent # 6,718,361, issued April 6, 2004. European Patent # EP1305924 issued March 31, 2010.

Distributed Routing, US Patent # 6,370,584, issued April 9, 2002.

Method and Apparatus for Assigning Tasks in a Distributed Server System, US Patent # 6,223,205, issued April 24, 2001.

Doctoral Students

Ahmed Youssef, Ph.D. June 2023. "Computational Methods to Uncover Cell State Proteomes and Profile Protein Interaction Dynamics." Co-advised with Andrew Emili.

Bashir Rastegarpanah, Ph.D. August 2021. "Tools for Responsible Decision-Marking in Machine Learning."

Larissa Spinelli, Ph.D. June 2019. "Empirical Studies of Factors Affecting Opinion Dynamics."

Giovanni Comarela, Ph.D. July 2017. "On the Dynamics of Interdomain Routing in the Internet."

Natali Ruchansky, Ph.D. August 2016. "Matrix Completion with Structure." Co-advised with Evimaria Terzi.

Andrej Cvetkovski, Ph.D. May 2013. "Graph Embeddings for Low-Stretch Greedy Routing."

Gonca Gürsun, Ph.D. March 2013. "Inferring Hidden Features in the Internet."

Vijay Erramilli, Ph.D. December 2008. "Forwarding in Mobile Opportunistic Networks."

Nahur Fonseca, Ph.D. December 2008. "Stochastic Modeling Applied to Detection Problems in Network Protocols and Traffic."

Anukool Lakhina, Ph.D. August 2006. "Network-Wide Traffic Analysis: Methods and Applications."

Jun Liu, Ph.D. August 2002. "Characterizing Network Elements and Paths Using Packet Loss Behavior."

Paul Barford, Ph.D. June 2000. "Modeling, Measurement and Performance of World Wide Web Transactions."

Robert L. Carter, Ph.D. January 1998. "Performance Measurement and Prediction in Packet-Switched Networks: Techniques and Applications."

Other Doctoral Students Advised

Mengfei Cao, Ph.D, August 2016, "New Graph Metrics Improve Network-based Protein Function Prediction" Tufts University, Member of the Committee.

Bimal Viswanath, Ph.D, April 2016, "Towards Trustworthy Social Computing Systems." Universität des Saarlandes, Member of the Committee.

Yonjun Liao, Ph.D., January 2013, "Learning to Predict End-to-End Network Performance." Université de Liège, Belgium, Member of the Jury.

Diana Zeaiter Joumblatt, Ph.D., December 2012, "Prediction of User Dissatisfaction with Internet Application Performance at End-Hosts." Université Pierre et Marie Curie, Rapporteur.

Chrisil Arackaparambil, Ph.D., July 2011, "Anomaly Detection in Network Streams Through a Distributional Lens." Dartmouth College, Member of the Examining Committee.

Italo Cunha, Ph.D., July 2011, "Tracking Internet Routes and Path Reachability for Network Tomography." Université Pierre et Marie Curie (U. Paris VI), Rapporteur.

Hung Nguyen, Ph.D., February 2008, "User-level Internet Tomography and Overlay Routing." Ecole Polytechnique Fédérale de Lausanne (EPFL), Rapporteur.

Benoit Donnet, Ph.D., September 2006, "Algorithms for Large-Scale Topology Discovery." Université Pierre et Marie Curie (U. Paris VI), Président du Jury.

David Chua, Ph.D. (Mathematics), May 2006, "Statistical Analysis for Whole Networks." Second Reader.

Stilian Stoev, Ph.D. (Mathematics) May 2005.

Abhishek Kumar, Ph.D., October 2005, "Network Data Streaming: Algorithms for Network Monitoring and Coordination." Georgia Tech, Committee Member.

Marjan Bozinovski, Ph.D., May 2004, "Fault-Tolerant Platforms for IP-Based Session Control Systems." Aalborg University, Committee Member.

Artur Ziviani, Ph.D. December 2003, "Qualité de Service et Localisation d'Hotes." University of Paris VI, Examinateur.

Liang Guo, Ph.D. April 2003, "Size-Aware Scheduling of TCP Flows." Second Reader.

Alberto Medina, Ph.D. March 2003, "Practical Estimation of Internet Traffic Demands." Third Reader.

Vladas Pipiras, Ph.D. (Mathematics) April 2002, "Symmetric Stable Self-Similar Processes." Third Reader.

Khaled A. Harfoush, Ph.D. November 2001, "A Framework for End-to-End Characterization of Metric-Induced Network Topologies." Committee Member.

Santiago M. Pericas-Geertsen, Ph.D. November 2001, "XML-Fluent Mobile Agents." Committee Member.

Alia K. Atlas, Ph.D. August 1998, "Statistical Rate Monotonic Scheduling." Second Reader.

Ghulam Bhatti, Ph.D. January 1998, "Parallel and Distributed Discrete Event Simulation: An Event-Reservation Approach." Second Reader.

Sulaiman Mirdad, Ph.D. June 1997, "WebWave: Globally Load-Balanced, Fully-Distributed Caching". Second Reader.

Gi Tae Kim, Ph.D. June 1997, "Framework for Adaptive Forward Erasure Recovery (AFER) for Fault-Tolerant Real-Time Communications". Second Reader.

Carlos Cunha, Ph.D. June 1997, "Trace Analysis and its Application to Performance Enhancements of Distributed Information Systems". Second Reader.

Susan Nagy, Ph.D. April 1997, "Admission Control and Scheduling Strategies for Real-Time Database Systems". Second Reader.

Vadim Teverovksy, Ph.D. February 1997 (Mathematics), "Detection and Estimation of Long-Range Dependence." Third Reader.

Euthimios Panagos, Ph.D. May 1995, "Client-Based Logging: A New Paradigm for Distributed Transaction Management". Third Reader.

Masters Students

Megan Hacker, M.S. May 2012, "Methods for Pairing Mobile Devices Through Shared Sensing"

Lauren Walters, MSc. May 2004, "A Web Browsing Workload Model for Simulation," University of Cape Town, South Africa, external examiner.

Alan Charbonneau, M.A. May 2003, "Simulation of an SJF Proxy"

Iana Vitkova, M.A. May 2003, "Simulation of an SJF Proxy"

Robert Frangioso, M.A. Sept 2002, "Connection Scheduling in Web Servers"

Xiaohui Zhang, M.S. May 1999, "Cachability of Web Objects"

Scott Harrison, M.S. May 1998, "DPR Monitor: A Web-based DPR Network Monitoring Tool"

Stanley Gambarin, M.S. May 1997, "Implementing a Single-Process Non-Multithreaded World Wide Web Server"

Kostas Magoutis, M.S. May 1996, "Economies of Scale In Partitioned Disk Caches with Self-Similar References"

Fred Bierhaus, M.S. May 1996, "Improving Interactive Traffic Through the EQL Driver"

Books

M. Crovella and B. Krishnamurthy, "Internet Measurement: Infrastructure, Traffic, and Applications," John Wiley and Sons, 2006, 512 pp.

BOOKS EDITED

M. Crovella and L. M. Feeney and D. Rubenstein (editors), "Proceedings of the 9th International IFIP TC 6 Networking Conference," Springer-Verlag, 410pp, 2010.

JOURNAL ARTICLES

Hui Peng, Sergei Kotelnikov, Megan E. Egbert, Shany Ofaim, Grant C. Stevens, Sadhna Phanse, Tatiana Saccon, Mikhail Ignatov, Shubham Dutta, Zoe Istace, Mohamed Taha Moutaoufik, Hiroyuki Aoki, Neal Kewalramani, Jianxian Sun, Yufeng Gong, Dzmitry Padhorny, Gennady Poda, Andrey Alekseenko, Kathryn A. Porter, George Jones, Irina Rodionova, Hongbo Guo, Oxana Pogoutse, Suprama Datta, Milton Saier, Mark Crovella, Sandor Vajda, Gabriel Moreno-Hagelsieb, John Parkinson, Daniel Segre, Mohan Babu, Dima Kozakov, and Andrew Emili, "Ligand interaction landscape of transcription factors and essential enzymes in *E. coli*," *Cell,* 2025.

A. Youssef, I. Paul, M. Crovella, and A. Emili, "DESP: Demixing Cell State Profiles from Dynamic Bulk Molecular Measurements," *Cell Reports Methods,* 2024.

L. Salamatian, S. Anderson, J. Matthews, P. Barford, W. Willinger, and M. Crovella, "A Manifold View of Connectivity in the Private Backbone Networks of Hyperscalers," *Communications of the ACM*, August 2023.

N. Kewalramani, A. Emili, M. Crovella, "State-of-the-Art Computational Methods to Predict Protein-Protein Interactions with High Accuracy and Coverage," *Proteomics,* 2023.

A. Youssef, F. Bian, P. Havugimana, M. Crovella, and A. Emili, "Dynamic Remodeling of *Escherichia coli* Interactome in Response to Environmental Perturbations," *Proteomics,* 2023.

L. Salamatian, S. Anderson, J. Mathews, P. Barford, W. Willinger, and M. Crovella, "A Manifold View of Connectivity in the Private Backbone Networks of Hyperscalers," *Communications of the ACM*, Research Highlights, 6(7), July 2023.

J. Jung, M. Guo, M. Crovella, J.G. McDaniel, and K.M. Warkentin. "Frog Embryos Use Multiple Levels of Temporal Pattern in Risk Assessment for Vibration-Cued Hatching," *Animal Cognition,* 2022.

J. N. Law, K. Akers, N. Tasnina, C. M. Della-Santina, S. Deutsch, M. Kshirsagar, J. Klein-Seetharaman, M. Crovella, P. Rajagopalan, S. Kasif, and T. M. Murali, "Interpretable Network Propagation with Application to Expanding the Repertoire of Human Proteins that Interact with SARS-CoV-2," *GigaScience,* 10(12), December 2021.

D. Lancour, J. Dupuis, R. Mayeux, J. Haines, M. Pericak-Vance, G. Schellenberg, M. Crovella, L. Farrer, and S. Kasif, "Analysis of Brain Region-Specific Co-Expression Networks Reveals Clustering of Established and Novel Genes Associated with Alzheimer Disease," *Alzheimer's Research & Therapy,* 2020.

J. Fan, X. C. Li, M. Crovella, and M. D. M. Leiserson, "Matrix (Factorization) Reloaded: Flexible Methods for Imputing Genetic Interactions with Cross-Species and Side Information," *OUP Bioinformatics,* 2020.

A. Ramirez, S. Dankel, B. Rastegarpanah, W. Cai, R. Xue, M. Crovella, Y.-H. Tseng, R. Kahn, and S. Kasif, "Single-Cell Transcriptional Networks in Differentiating Preadipocytes Suggest Drivers Associated with Tissue Heterogeneity," *Nature Communications,* 2020.

J. Fan, A. Cannistra, I. Fried, T. Lim, T. Schaffner, M. Crovella, B. Hescott, and M.D.M. Leiserson, "Functional Protein Representations from Biological Networks Enable Diverse Cross-Species Inference," *Nucleic Acids Research,* 2019.

D. Lancour, A. Naj, R. Mayeux, J. L. Haines, M. A. Pericak-Vance, G. C. Schellenberg, M. Crovella, L. A. Farrer, and S. Kasif, "One for All and All for One: Improving Replication of Genetic Studies through Network Diffusion," *PLoS Genetics*, Vol 14, April 2018.

A. Cvetkovski and M. Crovella, "Low-stress Data Embedding in the Hyperbolic Plane Using Multidimensional Scaling," *Applied Mathematics and Information Sciences,* Vol 11, Number 1, January 2017.

A. Cvetkovski and M. Crovella, "Multidimensional Scaling in the Poincaré Disk," *Applied Mathematics and Information Sciences,* Vol 10, Number 1, January 2016.

M. Cao, H. Zhang, J. Park, N. Daniels, M. Crovella, L. Cowen, and B. Hescott, "Going the Distance for Protein Function Prediction: A New Distance Metric for Protein Interaction Networks," *PLoS One,* Vol 8, Number 10, October 2013.

A. Cvetkovski and M. Crovella, "On the Choice of a Spanning Tree for Greedy Embedding of Network Graphs," *Networking Science,* February 2013.

B. Donnet, P. Raoult, T. Friedman, and M. Crovella, "Deployment of an Algorithm for Large-Scale Topology Discovery," *IEEE Journal on Selected Areas in Communications, Special Issue on Sampling the Internet*, Vol 24, Number 12, December 2006.

D. Chua, E. Kolaczyk, and M. Crovella, "Network Kriging," *IEEE Journal on Selected Areas in Communications, Special Issue on Sampling the Internet*, 2006.

B. Gueye, A. Ziviani, M. Crovella, and S. Fdida, "Constraint-Based Geolocation of Internet Hosts," *IEEE/ACM Transactions on Networking,* Vol 14, Number 6, December 2006.

R. Fonseca, V. Almeida, and M. E. Crovella, "Locality in a Web of Streams," *Communications of the ACM*, 48(1), pp. 82–88, January 2005.

M. Fayed, P. Krapivsky, J. Byers, M. Crovella, D. Finkel, and S. Redner, "On the Emergence of Highly Variable Distributions in the Autonomous System Topology," *Computer Communcation Review*, 33(2), pp. 41–49, July 2003.

A. Lakhina, J. W. Byers, M. E. Crovella, and I. Matta, "On the Geographic Location of Internet Resources," *IEEE Journal on Selected Areas in Communications, Special Issue on Internet and WWW Measurement, Mapping, and Modeling*, 2003.

P. Barford and M. E. Crovella, "Critical Path Analysis of TCP Transactions," *IEEE/ACM Transactions on Networking*, 9 (3), pp. 238–248.

M. E. Crovella, C. Lindemann, and M. Reiser, "Internet Performance Modeling: the state of the art at the turn of the century," *Performance Evaluation,* 42 (2–3), pp. 91-108, September 2000.

R. L. Carter and M. E. Crovella, "On the Network Impact of Dynamic Server Selection," *Computer Networks,* 31 (23–24), pp. 2529–2558.

M. Harchol-Balter, M. E. Crovella and C. D. Murta, "On Choosing a Task Assignment Policy for a Distributed Server System," *Journal of Parallel and Distributed Computing,* Special Issue on Software Support for Distributed Computing, September 1999.

P. Barford and M. E. Crovella, "Measuring Web Performance in the Wide Area," *Performance Evaluation Review,* Special Issue on Network Traffic Measurement and Workload Characterization, August 1999.

P. Barford, A. Bestavros, A. Bradley, and M. E. Crovella, "Changes in Web Client Access Patterns: Characteristics and Caching Implications," *World Wide Web,* Special Issue on World Wide Web Characterization and Performance Evaluation, 1999.

M. E. Crovella and M. S. Taqqu, "Estimating the Heavy Tail Index from Scaling Properties," *Methodology and Computing In Applied Probability,* 1(1), 1999.

M. E. Crovella and A. Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes," *IEEE/ACM Transactions on Networking,* 5(6):835–846, December 1997.

R. L. Carter and M. E. Crovella, "Measuring Bottleneck Link Speed in Packet-Switched Networks," *Performance Evaluation,* 27&28:297–318, October 1996.

L. A. Crowl, M. E. Crovella, T. J. LeBlanc, and M. L. Scott, "The Advantages of Multiple Parallelizations in Combinatorial Search," *Journal of Parallel and Distributed Computing,* special issue on Data-Parallel Algorithms and Programming, 21(1):110–123, April 1994.

Book Chapters

M. E. Crovella, M. S. Taqqu, and A. Bestavros, "Heavy-Tailed Probability Distributions in the World Wide Web," Chapter 1 of *A Practical Guide To Heavy Tails,* pages 3–25, Chapman & Hall, New York, 1998.

M. E. Crovella and L. Lipsky, "Simulations with Heavy-Tailed Workloads," Chapter 3 of *Self-Similar Network Traffic and Performance Evaluation,* Wiley/Wiley-Interscience, 1999.

K. Park, G. Kim and M. Crovella, "The Protocol Stack and its Modulation Effect on Self-Similar Traffic," Chapter 16 of *Self-Similar Network Traffic and Performance Evaluation,* Wiley/Wiley-Interscience, 1999.

M. E. Crovella, "Performance Characteristics of the World Wide Web," in *White Book on Performance Evaluation,* Günter Haring, Christoph Lindemann, and Martin Reiser, editors, Springer-Verlag, 1999.

Invited Papers

M. E. Crovella, "Performance Evaluation with Heavy Tailed Distributions," in *Lecture Notes in Computer Science 1786*, Proceedings of the 11th International Conference on Modelling Tools and Techniques for Computer and Communication System Performance Evaluation (TOOLS '2000), pp. 1-9, March 2000. Revised version appeared in *Job Scheduling Strategies for Parallel Processing : 7th International Workshop,* pp. 1-10, June 16, 2001.

Refereed Conference Papers

N. Tasnina, M. Crovella, and T. M. Murali, "Provenance Tracing in Network Diffusion Algorithms," *Proceedings of the Pacific Symposium on Biocomputing (PSB),* Hawaii, January, 2026.

G. Franco and M. Crovella, "Pinpointing Attention-Causal Communication in Language Models," *Proceedings of NeurIPS*, San Diego, CA, December 2025

S. Jasina, L. Salamatian, P. Barford, M. Crovella, and W. Willinger, "A Breath of Fresh Air: Visualizing How Networks 'Breathe'", *Proceedings of the 24th ACM Workshop on Hot Topics in Networking (HotNets 2025)*, College Park, MD, November, 2025.

S. Buxbaum, L. Tassis, L. Boschelli, D. Christenson. G. Comarela, M. Crovella, and M. Varia, "Privacy-Preserving Machine Learning on Web Browsing for Public Opinion," *Proceedings of The International Symposium on Cyber Security, Cryptology and Machine Learning (CSCML),* December, 2025.

P. Calais, G. Franco, Z. Tang, T. Nikas, W. Meira Jr., E. Terzi, and M. Crovella, "Disentangling Text and Math in Word Problems: Evidence for the Bidimensional Structure of Large Language Models' Reasoning," *Findings of the Association for Computational Linguistics,* June, 2025.

C. Kranig, E. Pauley, W.-S. Wang, P. Barford, M. Crovella, and J. Sommers, "Toward a Representative DNS Data Corpus: A Longitudinal Comparison of Collection Methods, *Proceedings of the Network Traffic Measurement and Analysis Conference,* Copenhagen, Denmark, June 2025. Recipient of the TMA Community Contribution Award.

W.-S. Wung, C. Kranig, E. Pauley, P. Barford, M. Crovella, and J. Sommers, "Squatspotting: Towards the Systematic Measurement of Typosquatting Techniques," *Proceedings of IFIP Networking,* Limassol, Cyprus, May 2025.

A. Anderson, A. S. Mondal, P. Barford, M. Crovella, and J. Sommers, "An Elemental Decomposition of DNS Name-to-IP Graphs," *Proceedings of INFOCOM*, Vancouver, Canada, May 2024.

G. Franco, M. Crovella, and G. Comarela, "Dependence and Model Selection in LLP: The Problem of Variants," *Proceedings of KDD*, Los Angeles, CA, August 2023.

K. Quinn, E. Terzi, and M. Crovella, "Characterizing Covid Waves via Spatio-Temporal Decomposition," *Proceedings of KDD*, Washington DC, August 2022.

L. Salamatian, S. Anderson, J. Matthews, P. Barford, W. Willinger, and M. Crovella, "Curvature-based Analysis of Network Connectivity in Private Backbone Infrastructures," *Proceedings of ACM SIGMETRICS / IFIP Performance,* Mumbai, India, June 2022.

B. Rastegarpanah, K. P. Gummadi, and M. Crovella, "Auditing Black-Box Prediction Models for Data Minimization Compliance," *Proceedings of NeurIPS,* Online, December 2021. **Selected as a Spotlight Presentation (3% acceptance rate).**

L. Amichi, A. Carneiro Viana, M. Crovella, and A. A. F. Loureiro, "From Motion Purpose to Perceptive Spatial Mobility Prediction," *Proceedings of ACM SIGSPATIAL*, Beijing, China, November 2021.

G. Grassi, C. Barakat, M. Crovella, and R. Teixeira, "Leveraging Website Popularity Differences to Identify Performance Anomalies," *Proceedings of Infocom,* Online, May 2021.

L. Amichi, A. Carneiro Viana, M. Crovella, and A. A. F. Loureiro, "Understanding Individuals' Proclivity for Novelty Seeking," *Proceedings of ACM SIGSPATIAL,* Online, November 2020.

J. Fan, X. C. Li, M. Crovella, and M. D. M. Leiserson, "Matrix (Factorization) Reloaded: Flexible Methods for Imputing Genetic Interactions with Cross-Species and Side Information," *Proceedings of the 19th European Conference on Computational Biology,* Online, August 2020.

B. Rastegarpanah, M. Crovella, and K. P. Gummadi, "Fair Inputs and Fair Outputs: the Incompatibility of Fairness in Privacy and Accuracy," *Proceedings of the Workshop on Fairness in User Modeling, Adaptatation, and Personalization (FairUMAP),* Online, July 2020.

L. Spinelli and M. Crovella, "How YouTube Leads Privacy-Seeking Users Away from Reliable Information," *Proceedings of the Workshop on Fairness in User Modeling, Adaptatation, and Personalization (FairUMAP),* Online, July 2020.

J. Law, M. Kshirsagar, M. Crovella, P. Rajagopalan, J. Klein-Seetharaman, S. Kasif and T. M. Murali, "A Network-Based Label Propagation Framework to Reposition Drugs for COVID-19 (Poster)," Proceedings of RECOMB, Online, June 2020.

L. Amichi, A. Viana, M. Crovella, and A. A. F. Loureiro, "Mobility profiling: Identifying Scouters in the Crowd (Poster)," *Proceedings of the CoNEXT Student Workshop,* Orlando, FL, December 2019.

B. Rastegarpanah, K. P. Gummadi, and M. Crovella, "Fighting Fire with Fire: Using Antidote Data to Improve Polarization and Fairness of Recommender Systems," *Proceedings of WSDM 2019,* Melbourne, Australia, February 2019.

G. Comarela, R. Durairajan, P. Barford, D. Christenson, and M. Crovella, "Assessing Candidate Preference through Web Browsing History," *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD),* London, UK, August 2018. Press coverage: BU Today: `http://www.bu.edu/today/2018/better-election-predictions`.

L. Spinelli and M. Crovella, "Unravelling the Dynamics of Online Ratings," Proceedings of the 4th International Symposium on Social Media Mining and Analysis (SMMA 2018), Exeter, UK, June 2018.

M.D.M. Leiserson, J. Fan, A. Cannistra, I. Fried, T. Lim, T. Schaffner, M. Crovella, and B. Hescott, "A Multi-Species Functional Embedding Integrating Sequence and Network Structure," Proceedings of RECOMB, Paris, France, April 2018.

B. Rastegarpanah, K. Gummadi, and M. Crovella, "Exploring Explanations for Matrix Factorization Recommender Systems" (Position Paper), FATREC Workshop on Responsible Recommendation, Como Italy, August 2017.

L. Spinelli and M. Crovella, "Closed-Loop Opinion Formation", *Proceedings of the 9th International ACM Web Science Conference (WebSci),* Troy, NY, June 2017.

N. Ruchansky, E. Terzi, and M. Crovella, "Targeted Matrix Completion," *Proceedings of the SIAM International Conference on Data Mining (SDM),* Houston, TX, April 2017.

C. Mavroforakis, D. Erdös, M. Crovella, and E. Terzi, "Active Positive-Definite Matrix Completion," *Proceedings of the SIAM International Conference on Data Mining (SDM),* Houston, TX, April 2017.

G. Comarela, E. Terzi, and M. Crovella, "Detecting Unusually-Routed ASes: Methods and Applications," *Proceedings of the ACM Internet Measurement Conference (IMC),* Santa Monica, CA, November 2016.

N. Ruchansky, M. Crovella, and E. Terzi, "Matrix Completion with Queries," *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD),* Sydney, Australia, August 2015.

G. Comarela and M. Crovella, "Identifying and Analyzing High Impact Routing Events with PathMiner," *Proceedings of the ACM Internet Measurement Conference (IMC),* Vancouver, BC, November 2014.

B. Viswanath, M. A. Bashir, M. Crovella, S. Guha, K. Gummadi, B. Krishnamurthy, and A. Mislove, "Towards Detecting Anomalous User Behavior in Online Social Networks," *Proceedings of the 23rd USENIX Security Symposium*, San Diego, CA, August 2014.

Y. Zhang, T. Lappas, M. Crovella, and E. Kolaczyk, "Online ratings: convergence towards a positive perspective?," *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP),* Florence, Italy, May 2014.

G. Comarela, G. Gürsun, and M. Crovella, "Studying Interdomain Routing over Long Timescales," *Proceedings of the ACM Internet Measurement Conference (IMC),* Barcelona, Spain, October 2013.

L. Spinelli, M. Crovella and B. Eriksson, "AliasCluster: A Lightweight Approach to Interface Disambiguation," *Proceedings of the Global Internet Symposium,* Turin, Italy, April 2013.

B. Eriksson and M. Crovella, "Understanding Geolocation Accuracy using Network Geometry," in *Proceedings of the Infocom 2013 Miniconference,* Turin, Italy, April 2013.

J. M. Agosta, J. Chandrasekar, M. Crovella, N. Taft, and D. Ting, "Mixture Models of Endhost Network Traffic," in *Proceedings of the Infocom 2013 Miniconference,* Turin, Italy, April 2013.

G. Gürsun, N. Ruchansky, E. Terzi, and M. Crovella, "Routing State Distance: A Path-based Metric for Network Analysis," in *Proceedings of the ACM Internet Measurement Conference (IMC),* Boston, MA, November 2012. **Winner of a 2013 IETF/IRTF Applied Networking Research Prize.**

G. Gürsun and M. Crovella, "On Traffic Matrix Completion in the Internet," in *Proceedings of the ACM Internet Measurement Conference (IMC),* Boston, MA, November 2012.

T. Lappas, M. Crovella, and E. Terzi, "Selecting a Characteristic Set of Reviews," in *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD) 2012,* Beijing, China, August 2012.

Q. Ding, N. Katenka, P. Barford, E. Kolaczyk, and M. Crovella, "Intrusion as (Anti)social Communication: Characterization and Detection," in *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD) 2012,* Beijing, China, August 2012. **Best Student Paper Award.**

G. Gürsun, N. Ruchansky, E. Terzi, and M. Crovella, "Inferring Visibility: Who's (not) Talking to Whom?," in *Proceedings of SIGCOMM 2012,* Helsinki, Finland, August 2012.

B. Eriksson and M. Crovella, "Estimating Intrinsic Dimension Via Clustering," in *IEEE Statistical Signal Processing Workshop (SSP),* Ann Arbor, MI, August 2012.

C. Thomas, J. Sommers, P. Barford, D. Kim, A. Das, R. Segebre, M. Crovella, "A Passive Measurement System for Network Testbeds," in *Tridentcom 2012,* Thessaloniki, Greece, June 2012.

G. Comarela, M. Crovella, V. Almeida and F. Benevenuto, "Understanding Factors that Affect Response Rates in Twitter," in *ACM Hypertext 2012,* Milwaukee, WI, June 2012.

A. Cvetkovski and M. Crovella, "Low-Stretch Greedy Embedding Heuristics," in *Fourth International Workshop on Network Science for Communication Networks (NetSciCom 2012),* Orlando, FL, March 2012.

V. Manfredi, M. Crovella, and J. Kurose, "Understanding Stateful vs Stateless Communication Strategies for Ad hoc Networks," in *Proceedings of Mobicom 2011,* Las Vegas, NV, September 2011.

G. Gürsun, M. Crovella, and I. Matta, "Describing and Forecasting Video Access Patterns," in *Proceedings of the Infocom 2011 Miniconference,* Shanghai, China, April 2011.

V. Bharti, P. Kankar, L. Setia, G. Gürsun, and M. Crovella, "Inferring Invisible Traffic," in *Proceedings of CoNEXT 2010,* Philadelphia, PA, December 2010.

J. Sommers, P. Barford and M. Crovella, "Router Primitives for Programmable Active Measurement," in *Workshop on Programmable Routers for Extensible Services of Tomorrow (PRESTO '09)*, Barcelona, Spain, August, 2009.

A. Cvetkovski and M. Crovella, "Hyperbolic Embedding and Routing for Dynamic Graphs," in *Proceedings of Infocom 2009*, Rio de Janeiro, Brazil, April 2009.

V. Erramilli and M. Crovella, "Forwarding in Opportunistic Networks with Resource Constraints," in *Proceedings of the Fourth ACM Workshop on Challenged Networks (CHANTS 08)*, San Francisco, CA, September, 2008.

N. Fonseca, M. Crovella, and K. Salamatian, "Long Range Mutual Information," in *Proceedings of the First Workshop on Hot Topics in Measurement and Modeling of Computer Systems (Hotmetrics '08),* Annapolis, MD, June, 2008.

V. Erramilli, A. Chaintreau, M. Crovella, and C. Diot, "Delegation Forwarding," in *Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing (ACM MobiHoc),* Hong Kong SAR, China, May 2008.

P. Chhabra, C. Scott, E. D. Kolaczyk, and M. Crovella, "Distributed Spatial Anomaly Detection," in *Proceedings of Infocom 2008*, Phoenix, AZ, April 2008.

V. Erramilli, A. Chaintreau, M. Crovella, and C. Diot, "Diversity of Forwarding Paths in Pocket Switched Networks," in *Proceedings of the ACM SIGCOMM Internet Measurement Conference*, San Diego, October, 2007.

B. Eriksson, P. Barford, R. Nowak, and M. Crovella, "Learning Network Structure from Passive Measurements," in *Proceedings of the ACM SIGCOMM Internet Measurement Conference*, San Diego, October, 2007.

V. Erramilli, M. Crovella, and N. Taft, "An Independent-Connection Model for Traffic Matrices," in *Proceedings of the ACM SIGCOMM Internet Measurement Conference*, Rio de Janeiro, October, 2006.

X. Li, F. Bian, M. Crovella, C. Diot, R. Govindan, G. Iannaccone, and A. Lakhina, "Detection and Identification of Network Anomalies Using Sketch Subspaces," in *Proceedings of the ACM SIGCOMM Internet Measurement Conference*, Rio de Janeiro, October, 2006.

A. Lakhina, M. Crovella, and C. Diot, "Detecting Distributed Attacks using Network-Wide Flow Traffic," in *Proceedings of FloCon 2005 Analysis Workshop*, New Orleans, LA, September, 2005.

A. Lakhina, M. Crovella, and C. Diot, "Mining Anomalies Using Traffic Feature Distributions," in *Proceedings of ACM SIGCOMM*, Philadelphia, PA, August 2005.

D. B. Chua, E. D. Kolaczyk, and M. Crovella, "A Statistical Framework for Efficient Monitoring of End-to-End Network Properties," (Poster) in *Proceedings of ACM SIGMETRICS*, Banff, Canada, June 2005.

B. Donnet, P. Raoult, T. Friedman, and M. Crovella, "Efficient Algorithms for Large-Scale Topology Discovery," in *Proceedings of ACM SIGMETRICS*, Banff, Canada, June 2005.

A. Soule, A. Lakhina, N. Taft, K. Papagiannaki, K. Salamatian, A. Nucci, M. Crovella, and C. Diot, "Traffic Matrices: Balancing Measurements, Inference and Modeling," in *Proceedings of ACM SIGMETRICS*, Banff, Canada, June 2005.

B. Donnet, T. Friedman, and M. Crovella, "Improved Algorithms for Network Topology Discovery," in *Lecture Notes in Computer Science, Proceedings of the 2005 Passive and Active Measurement Workshop*, Boston, MA, March, 2005.

D. B. Chua, E. D. Kolaczyk, and M. Crovella, "Efficient Monitoring of End-to-End Network Properties," in *Proceedings of Infocom 2005*, Miami, FL, March 2005.

N. Fonseca and M. Crovella, "Bayesian Packet Loss Detection for TCP," in *Proceedings of Infocom 2005*, Miami, FL, March 2005.

B. Gueye, A. Ziviani, M. Crovella, and S. Fdida, "Constraint-Based Geolocation of Internet Hosts," in *Proceedings of the ACM SIGCOMM Internet Measurement Conference*, Taormina, Italy, October 2004.

A. Lakhina, M. Crovella, and C. Diot, "Characterization of Network-Wide Anomalies in Traffic Flows," in *Proceedings of the ACM SIGCOMM Internet Measurement Conference*, Taormina, Italy, October 2004.

A. Lakhina, M. Crovella, and C. Diot, "Diagnosing Network-Wide Traffic Anomalies," in *Proceedings of ACM SIGCOMM*, Portland, OR, August 2004.

A. Lakhina, M. Crovella, and C. Diot, "Exploring the Subspace Method for Network-Wide Anomaly Diagnosis," (Poster) in *Network Troubleshooting Workshop: Research, Theory and Operations*, Portland, OR, August 2004.

A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. D. Kolaczyk, and N. Taft, "Structural Analysis of Network Traffic Flows," in *Proceedings of ACM SIGMETRICS / Performance 2004*, New York, NY, June 2004.

L. Tang and M. E. Crovella, "A Geometric Exploration of the Landmark Selection Problem," in *Lecture Notes in Computer Science 3015, Proceedings of the 2004 Passive and Active Measurement Workshop (PAM2004)*, pp. 63–72, Juan-les-Pins, France, April 2004.

L. Tang and M. E. Crovella, "Virtual Landmarks for the Internet," in *Proceedings of the 2003 Internet Measurement Conference*, Miami, Florida, October 2003.

R. Fonseca, V. Almeida, M. E. Crovella, and B. Abrahão, "On the Intrinsic Locality Properties of Web Reference Streams," in *Proceedings of IEEE Infocom 2003,* San Francisco, California, April 2003.

A. Lakhina, J. Byers, M. E. Crovella, and P. Xie, "Sampling Biases in IP Topology Measurements," in *Proceedings of IEEE Infocom 2003,* San Francisco, California, April 2003.

M. E. Crovella and E. Kolaczyk, "Graph Wavelets for Spatial Traffic Analysis," in *Proceedings of IEEE Infocom 2003,* San Francisco, California, April 2003.

J. Liu, I. Matta, and M. E. Crovella, "End-to-End Inference of Loss Nature in a Hybrid Wired/Wireless Environment," in *Proceedings of Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt'03)*, Sophia-Antipolis, France, March 2003.

A. Lakhina, J. W. Byers, M. E. Crovella, and Ibrahim Matta, "On the Geographic Location of Internet Resources," (Extended Abstract) in *Proceedings of the ACM SIGCOMM Internet Measurement Workshop 2002,* Marseilles, France, November 2002.

J. Liu and M. Crovella, "Using Loss Pairs to Discover Network Properties," in *Proceedings of the ACM SIGCOMM Internet Measurement Workshop 2001,* pp. 127–138, San Francisco, CA, November 2001.

P. Barford, A. Bestavros, J. Byers, and M. Crovella, "On the Marginal Utility of Network Topology Measurements," in *Proceedings of the ACM SIGCOMM Internet Measurement Workshop 2001,* pp. 5–17, San Francisco, CA, November 2001.

L. Guo, I. Matta, and M. E. Crovella, "How Does TCP Generate Pseudo-Self-Similarity?," in *Proceedings of MASCOTS 2001*, Cincinnati, Ohio, August 2001.

P. Barford and M. E. Crovella, "Critical Path Analysis of TCP Transactions," in *Proceedings of the 2000 ACM SIGCOMM Conference*, Stockholm, Sweden, pp. 127–138, September 2000. Also appeared by invitation as "Análisis de secuencia crítica de transacciones TCP" in *Proceedings of SIGCOMM Latin America,* San Antonio de Belén, Costa Rica, April 2001.

M. E. Crovella, R. Frangioso, and M. Harchol-Balter "Connection Scheduling in Web Servers," in *1999 USENIX Symposium on Internet Technologies and Systems (USITS '99).*

P. Barford and M. E. Crovella, "A Performance Evaluation of Hyper Text Transfer Protocols," in *Proceedings of ACM SIGMETRICS '99,* pp. 188–197, Atlanta, Georgia, May 1999.

A. Bestavros, M. E. Crovella, J. Liu, and D. Martin, "Distributed Packet Rewriting and its Application to Scalable Server Architectures," in *Proceedings of the Sixth International Conference on Network Protocols (ICNP '98),* pp. 290–297, October, 1998.

M. Harchol-Balter, M. E. Crovella, and C. D. Murta, "On Choosing a Task Assignment Policy for a Distributed Server System," in *Proceedings of Performance Tools '98, Lecture Notes in Computer Science,* No. 1469, pp. 231–242, September 1998.

P. Barford and M. E. Crovella, "Generating Representative Web Workloads for Network and Server Performance Evaluation,"in *Proceedings of ACM SIGMETRICS/Performance '98,* pp. 151–160, Madison, Wisconsin, July 1998.

M. E. Crovella, M. Harchol-Balter, and C. D. Murta, "Task Assignment in a Distributed System: Improving Performance by Unbalancing Load," in *Proceedings of SIGMETRICS '98 (Poster paper),* Madison, Wisconsin, July 1998.

M. E. Crovella and P. Barford, "The Network Effects of Prefetching," in *Proceedings of IEEE Infocom '98, the Seventeenth Annual Joint Conference of the IEEE Computer and Communication Societies,* San Francisco, California, pp. 1232–1240, April 1998.

M. E. Crovella and L. Lipsky, "Long-Lasting Transient Conditions in Simulations with Heavy-Tailed Workloads," in *Proceedings of the 1997 Winter Simulation Conference*, December 1997.

K. Park, G. Kim, and M. E. Crovella, "On the Effect of Traffic Self-Similarity on Network Performance," in *Proceedings of the SPIE International Conference on Performance and Control of Network Systems*, November, 1997.

P. Fiorini, L. Lipsky, and M. E. Crovella, "Consequences of Ignoring Self-Similar Data Traffic In Communications Modeling," In *Tenth International Conference on Parallel and Distributed Computing Systems (PDCS-97),* New Orleans, LA, pp. 322-327, October 1997.

R. L. Carter and M. E. Crovella, "Server Selection using Dynamic Path Characterization in Wide-Area Networks," in *Proceedings of IEEE Infocom '97, the Sixteenth Annual Joint Conference of the IEEE Computer and Communication Societies,* Kobe, Japan, April 1997.

V. Almeida, A. Bestavros, M. Crovella, and A. de Oliveira, "Characterizing Reference Locality in the WWW," in *Proceedings of the Fourth International Conference on Parallel and Distributed Information Systems (PDIS '96)*, December 1996.

K. Park, G. Kim, and M. E. Crovella, "On the Relationship Between File Sizes, Transport Protocols, and Self-Similar Network Traffic," in *Proceedings of the Fourth International Conference on Network Protocols (ICNP '96),* pp. 171–180, October, 1996.

R. L. Carter and M. E. Crovella, "Measuring Bottleneck Link Speed in Packet-Switched Networks," in *PERFORMANCE '96, the International Conference on Performance Theory, Measurement and Evaluation of Computer and Communication Systems,* October 1996.

M. E. Crovella and A. Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes," in *Proceedings of the 1996 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems,* May 1996. **Winner of a 2010 ACM SIGMETRICS Test of Time Award.**

M. E. Crovella and R. L. Carter, "Dynamic Server Selection in the Internet," In *Proceedings of the Third IEEE Workshop on the Architecture and Implementation of High Performance Communication Subsystems (HPCS '95),* August 1995.

A. Bestavros, R. L. Carter, M. E. Crovella, C. R. Cunha, A. Heddaya, and S. A. Mirdad, "Application-Level Document Caching in the Internet," In *Proceeedings of the Second Workshop on Services in Distributed and Networked Environments (SDNE '95),* June 1995.

M. E. Crovella and T. J. LeBlanc, "Parallel Performance Prediction Using Lost Cycles Analysis," In *Proceedings of Supercomputing '94,* November, 1994.

R. Bianchini, M. E. Crovella, L. Kontothanassis, and T. J. LeBlanc, "Software Interleaving," In *Proceedings of the 6th IEEE Symposium on Parallel and Distributed Processing*, October, 1994.

M. E. Crovella and T. J. LeBlanc, "Performance Debugging using Parallel Performance Predicates," In *3rd ACM/ONR Workshop on Parallel and Distributed Debugging*, May, 1993, pp. 140-150.

M. Crovella, R. Bianchini, T. LeBlanc, E. Markatos, and R. Wisniewski, "Using Communication-to-Computation Ratio in Parallel Performance Prediction and Program Design," *Proceeedings of the Fourth IEEE Symposium on Parallel and Distributed Processing*, December 1992, pp. 238-245.

M. Crovella, P. Das, C. Dubnicki, T. LeBlanc, and E. Markatos, "Multiprogramming on Multiprocessors," *Proceeedings of the Third IEEE Symposium on Parallel and Distributed Processing*, December 1991, pp. 590–597, Also published as TR 385, Department of Computer Science, University of Rochester, February 1991 (revised May 1991).

E. Markatos, M. Crovella, P. Das, C. Dubnicki, and T. LeBlanc, "The Effects of Multiprogramming on Barrier Synchronization," *Proceeedings of the Third IEEE Symposium on Parallel and Distributed Processing*, December 1991, pp. 662-669. Also TR 380, University of Rochester Computer Science Department, May 1991.

N. Bashir, M. Crovella, G. DeTitta, F. Han, H. Hauptman, J. Horvath, H. King, D. Langs, R. Miller, T. Sabin, P. Thuman and D. Velmurugan, "Parallel Solutions to the Phase Problem in X-Ray Crystallography," *Proceedings of The Fifth Distributed Memory Computing Conference*, IEEE Computer Society Press, 1990, pp. 513–521.

M. Crovella, M. S. Summers, E. M. Camardo, and S. S. Weissman, "Exploiting a Restricted Application Domain in SDE Design," *Advance Working Papers of the Second International Workshop on Computer Aided Software Engineering,* IEEE Computer Society Press, July 1988, pp. 2-3–2-4.

J. G. Neal and M. E. Crovella, "An Expert System for Tactical Support Jammer Configuration," *Proceedings of the 1987 IEEE National Aerospace and Electronics Conference (NAECON '87),* May 1987, pp. 1308–1314.

J. G. Neal, M. E. Crovella, and C. Y. Thielman, "A Knowledge-Based Approach to an EW Mission Planning Task," *Proceedings of the Knowledge Based Information Systems Symposium (KBIS-87),* SHAPE Technical Centre, The Hague, Netherlands, 1987.

J. G. Neal, D. F. Funke, and M. E. Crovella, "JAMEX: An Expert Advisor on Tactical Support Jammer Configuration," *Proceedings of the Second Annual Conference on Expert Systems in Government,* IEEE Computer Society Press, November 1986, pp. 383–394.

Refereed Presentations

I. Fried, A. Cannistra, C. Casey, A. Piel, T. Lim, M. Crovella, B. Hescott, "CANDL: Coarsely Aligning Networks with Diffusion and Landmarks," Theme Presentation, ISMB/ECCB, Dublin, July 2015.

UNREFEREED PUBLICATIONS

R. Canetti, A. Cohen, C. Conley, M. Crovella, S. L. Dogan, M. Gaboardi, W. Hartzog, R. Van Loo, C. T. Robertson, K. Baird Silbaugh, "National Telecommunications and Information Administration: Comments from Researchers at Boston University and the University of Chicago," Boston Univ. School of Law Research Paper No. 23-24, June 2023.

M. Crovella, "The Skillful Interrogation of the Internet," in *Computer Communication Review,* vol. 49, no. 3, October 2019.

kc Claffy, M. Crovella, T. Friedman, C. Shannon and N. Spring, "Community-oriented network measurement infrastructure (CONMI) workshop report," in *Computer Communication Review,* vol. 36, no. 2, pp. 41–48, April, 2006.

M. E. Crovella, "How Big is the Average Web Page?" in *Inclusions, the Arts of Mathematics,* Journal of the Mathematical Association of America Student Chapter at Boston University, May 1999.

T. J. LeBlanc, M. L. Scott, B. Marsh, E. Markatos, C. Dubnicki, M. Crovella, and T. Becker, "The Psyche Parallel Operating System," *IEEE TC on Operating Systems Newsletter,* vol. 6, no. 1, 1992, pp. 11-13.

TECHNICAL REPORTS

G. Grassi, C. Barakat, M. Crovella, and R. Teixeira, "Leveraging Website Popularity Differences to Identify Performance Anomalies," (Extended Version), Available at `https://hal.inria.fr/hal-03109717`, January 2021.

J. N. Law, N. Tasnina, M. Kshirsagar, J. Klein-Seetharaman, M. Crovella, P. Rajagopalan, S. Kasif, and T. M. Murali, "Identifying Human Interactors of SARS-CoV-2 Proteins and Drug Targets for COVID-19 using Network-Based Label Propagation," arXiv:2006.01968, June 2, 2020.

B. Rastegarpanah, M. Crovella, and K. P. Gummadi, "Fair Inputs and Fair Outputs: the Incompatibility of Fairness in Privacy and Accuracy," arXiv:2005.09209, May 19, 2020.

M.D.M. Leiserson, J. Fan, A. Cannistra, I. Fried, T. Lim, T. Schaffner, M. Crovella, and B. Hescott, "A Multi-Species Functional Embedding Integrating Sequence and Network Structure," bioRxiv, December 4, 2017.

J.-M. Agosta, J. Chandrasekar, M. Crovella, N. Taft, and D. Ting, "Mixture Models of Endhost Network Traffic," arxiv:1212.2744, December 13, 2012.

M. Hacker, M. Crovella, and L. Reyzin, "Secure Pairing of Mobile Devices," BUCS-TR-2012-011, May 16, 2012.

A. Cvetkovski and M. Crovella, "Multidimensional Scaling in the Poincare Disk," arxiv:1105.5332, May 26, 2011.

A. Lakhina, M. Crovella, and C. Diot, "Diagnosing Network-Wide Traffic Anomalies," BUCS-TR-2004-008, February 24, 2004. Also published as a Sprint Research Report.

A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. D. Kolaczyk, and N. Taft, "Structural Analysis of Network Traffic Flows," BUCS-TR-2003-021, November 10, 2003. Also published as Sprint Research Report RR03-ATL-110708.

M. Fayed, P. Krapivsky, J. Byers, M. Crovella, and D. Finkel, "On the Size Distribution of Autonomous Systems," BUCS-TR-2003-001, January 17, 2003.

R. Fonseca, V. Almeida, M. E. Crovella, and B. Abrahão, "On the Intrinsic Locality Properties of Web Reference Streams," BUCS-TR-2002-022, July 15, 2002.

A. Lakhina, J. Byers, M. E. Crovella, and P. Xie, "Sampling Biases in IP Topology Measurements," BUCS-TR-2002-021, July 15, 2002.

M. E. Crovella and E. Kolaczyk, "Graph Wavelets for Spatial Traffic Analysis," BUCS-TR-2002-020, July 15, 2002.

A. Lakhina, J. W. Byers, M. E. Crovella, and Ibrahim Matta, "On the Geographic Location of Internet Resources," BUCS-TR-2002-015, 2002.

J. Liu, I. Matta, and M. E. Crovella, "End-to-End Inference of Loss Nature in a Hybrid Wired/Wireless Environment," BUCS-TR-2002-008, March 14, 2002.

M. E. Crovella, R. Frangioso, and M. Harchol-Balter "Connection Scheduling in Web Servers," BUCS-TR-1999-003, March 31, 1999.

M. Harchol-Balter, M. E. Crovella, and C. D. Murta, "On Choosing a Task Assignment Policy for a Distributed Server System," MIT Laboratory for Computer Science, MIT-LCS-TR-757, 1998.

P. Barford and M. E. Crovella, "Generating Representative Web Workloads for Network and Server Performance Evaluation," BUCS-TR-1997-006, November 4, 1997.

M. E. Crovella, M. Harchol-Balter, and C. D. Murta, "Task Assignment in a Distributed System: Improving Performance by Unbalancing Load," BUCS-TR-1997-018, October 31, 1997.

M. Harchol-Balter, M. E. Crovella, and C. D. Murta, "To Queue or Not to Queue: When Queueing is Better Than Timesharing in a Distributed System," BUCS-TR-1997-017, October 31, 1997.

K. Park, G. Kim, and M. E. Crovella, "On the Effect of Traffic Self-Similarity on Network Performance," CSD-TR 97-024, Dept. of Computer Sciences, Purdue University, July 1997.

M. Crovella and P. Barford, "The Network Effects of Prefetching," BUCS-TR-97-002, February 7, 1997.

K. Park, G. Kim, and M. E. Crovella, "On the Relationship Between File Sizes, Transport Protocols, and Self-Similar Network Traffic," BUCS-TR-96-016, August 7, 1996.

V. Almeida, A. Bestavros, M. Crovella, and A. deOliveira, "Characterizing Reference Locality in the WWW," BUCS-TR-96-011, June 21, 1996.

A. Bestavros, M. Chen, M. Crovella, A. Heddaya, S. Sclaroff, and J. Cowie, "Responsive Web Computing: Resource Management, Protocol Techniques, and Applications (A research statement)," BUCS-TR-96-008, March 21, 1996.

R. L. Carter and M. E. Crovella, "Dynamic Server Selection Using Bandwidth Probing in Wide-Area Networks," BUCS-TR-96-007, Boston University Computer Science Department, March 18, 1996.

R. L. Carter and M. E. Crovella, "Measuring Bottleneck Link Speed in Packet-Switched Networks," Technical Report BUCS-TR-96-006, Boston University Computer Science Department, March 15, 1996.

M. E. Crovella and A. Bestavros, "Explaining World Wide Web Traffic Self-Similarity," Technical Report BUCS-TR-95-015 (Revised), Boston University Department of Computer Science, October 1995.

M. E. Crovella, T. J. LeBlanc, and W. Meira Jr., "Performance Measurement and Modeling with the Lost Cycles Toolkit," Technical Report 580, University of Rochester Computer Science Department, June 1995.

C. Cunha, A. Bestavros, and M. Crovella, "Characteristics of WWW Client-based Traces," Technical Report BUCS-TR-95-010, Boston University Department of Computer Science, April 1995.

A. Bestavros, R. L. Carter, M. E. Crovella, C. R. Cunha, A. Heddaya, and S. A. Mirdad, "Application-Level Document Caching in the Internet," Technical Report BUCS-TR-95-002, Boston University Department of Computer Science, February 1995.

M. E. Crovella, "Performance Prediction and Tuning of Parallel Programs," Technical Report 573 and Ph.D. Thesis, Computer Science Dept., U. Rochester, February 1995.

M. E. Crovella and T. J. LeBlanc, "The Search for Lost Cycles: a New Approach to Parallel Program Performance Evaluation," Technical Report 479, University of Rochester Computer Science Department, December 1993.

L. A. Crowl, M. E. Crovella, T. J. LeBlanc, and M. L. Scott, "Beyond Data Parallelism: The Advantages of Multiple Parallelizations in Combinatorial Search," Technical Report 92-80-06, December 1992, Department of Computer Science, Oregon State University. Revised version appears as Technical Report 451, U. Rochester Department of Computer Science, April 1993.

R. Bianchini, M. E. Crovella, L. Kontothanassis, and T. J. LeBlanc, "Alleviating Memory Contention in Matrix Computations on Large-Scale Shared-Memory Multiprocessors," Technical Report 449, U. Rochester Computer Science Department, April 1993.

R. Bianchini, M. E. Crovella, L. Kontothanassis, and T. J. LeBlanc, "Memory Contention in Scalable Cache-Coherent Multiprocessors," Technical Report 448, U. Rochester Computer Science Department, April 1993.

D. Bergmark and M. Crovella, "Case Study in KSR Programming: Finding Outliers by the Minimum Volume Ellipsoid Method," Technical Report CTC92TR107, October 1992, Center for Theory and Simulation in Science and Engineering, Cornell University, Ithaca, NY 14853.

Recent Talks

"Beyond accuracy: understanding the performance of LLMs on exams designed for humans,"

- UFMG, Department of Computer Science, March 8, 2024.

"Estimating Traffic Flows that Can't Be Measured"

- Google Telemetry Seminar Series, May 17, 2021.

"Assessing Candidate Preference Through Web Browsing History"

- Sorbonne Université (UPMC) LIP6, April 15, 2019.
- INRIA Sophia-Antipolis, March 20, 2019.
- INRIA Saclay, December 13, 2018.
- INRIA Paris, November 8, 2018.
- LINCS Laboratory, Paris, October 10, 2018.

"Human Behavior is Low Dimensional"

- TMA Experts Summit 2019, Paris, June 18, 2019.
- Chinese Academy of Sciences, Institute of Computing Technology, Beijing, May 20, 2019.
- Huawei Network AI Workshop, Paris, April 9, 2019.
- LINCS Laboratory, Paris, February 13, 2019.
- Wesleyan University Department of Mathematics and Computer Science, April 13, 2017.
- IEEE Workshop on Data Science for Networking (DS4N), December 14, 2017.

"The Statistics of Human Activity and Implications for Security"

- BU Statistics Club, Professor Speaker Series, November 10, 2016.

"Analyzing the Evolution of Internet Routing through Path Analysis"

- Cisco FAST Seminar Series, July 17, 2015.
- Raytheon BBN Technologies, SDP Seminar, September 28, 2015.

"Reversing the Curse of Dimensionality for Security Analysis"

- NYU Poly Department of Computer Science, December 5, 2014.

"Anomaly Detection and Dimensionality Reduction"

- Workshop on Mathematics of Data Analysis in Cybersecurity, ICERM, Brown University, October 24, 2014.

"Targeted Matrix Completion"

- 51st Annual Allerton Conference on Communication, Control, and Computing, University of Illinois at Urbana-Champaign, October 3, 2013.

"Better Routing with Hyperbolic Geometry"

- Laboratory of Information, Networking and Communication Sciences (LINCS) Paris, December 19, 2012.

"A Fine Grained Distance Metric for Small Worlds"

- Université de Liège, Liège Belgium, January 11, 2013.
- Laboratory of Information, Networking and Communication Sciences (LINCS) Paris, Mar 20, 2012.

"Inferring Invisible Traffic"

- Williams College, Dept. of Computer Science, Williamstown, MA, November 22, 2013.
- Tufts University, Dept. of Computer Science, Medford, MA, November 7, 2013.
- University of Miami, Dept. of Computer Science, Miami, FL, March 2, 2011.
- International Computer Science Institute (ICSI), Berkeley, CA, July 20, 2010.
- Statistics of Networks Workshop, Isaac Newton Institute for Mathematical Sciences, Cambridge, UK, June 24, 2010.

"Network Monitoring: Challenges and Opportunities"

- Indian Institute of Technology (IIT), Delhi, India, June 9, 2008.

"Exploiting Feature Distributions in Anomaly Diagnosis"

- Dartmouth, November 28, 2005.
- MIT CSAIL-LIDS Networking and Systems Seminar, Cambridge MA, April 11, 2005.

"Network Anomaly Diagnosis"

- Boston Area Networking and Networkers Annual Summit (BANANAS-2005), BBN Technologies, Cambridge MA, January 20, 2005.

"Applying the Subspace Method to Network Traffic Analysis"

- Statistical Research Center for Complex Systems (SRCCS), Workshop on "Internet Measurement, Modeling, and Analysis," Seoul National University, Seoul, South Korea, January 11, 2005.
- Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea, January 13, 2005.

"Diagnosing Network-Wide Traffic Anomalies"

- Columbia University, Nov. 17, 2004.
- Workshop on Internet Signal Processing (WISP), CAIDA, Nov. 11, 2004.
- Ecole Normale Supérieure, Paris, Apr. 13, 2004.
- France Telecom R&D, Paris, Mar. 18, 2004.

"Structural Analysis of Network Traffic Flows"

- Project Metropolis Meeting, École Nationale Supérieure des Télécommunications de Bretagne (ENST), Rennes, Feb. 10, 2004.

- Institute for Mathematics and its Applications, Workshop on "Measurement, Modeling and Analysis of the Internet," Minneapolis, MN, January 12, 2004.

"Virtual Landmarks for the Internet"

- Intel Research, Cambridge UK, October 1, 2003.
- Ecole Polytechnique Fédérale de Lausanne (EPFL), November 10, 2003.
- Laboratoire d'Informatique de Paris VI (LIP6), November 14, 2003.

"Sampling Biases in IP Topology Measurements"

- Séminaire Graphes, Réseaux et Modélisation, Laboratoire de Recherche en Informatique, Université d'Orsay, January 8, 2004.
- IBM T.J. Watson Labs, April 11, 2003.
- BU Statistics Colloquium, April 10, 2003.
- National Institute of Statistical Sciences, March 28, 2003.