

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

PALO ALTO NETWORKS, INC.,
Petitioner,

v.

QOMPLX LLC,
Patent Owner.

Case IPR2026-00371
Patent 12,143,425

PATENT OWNER'S PRELIMINARY RESPONSE

TABLE OF CONTENTS

I.	INTRODUCTION	1
II.	THE PETITION FAILS TO DEMONSTRATE A REASONABLE LIKELIHOOD OF SUCCESS.	4
A.	Petitioner Does Not Show That Its Identified “first data” Represents A Portion Of A Distributed Computational Graph (All Grounds, All Claims).	4
1.	Petitioner Does Not Show That Barsness’s Monitoring Application “represents a portion of a distributed computational graph” (Ground 1, All Claims).	5
2.	Petitioner’s Multiple-Reference Grounds Do Not Cure This Defect Of Its Single-Reference Barsness Ground (Grounds 2-4, All Claims).	13
B.	Petitioner Does Not Show That Its Identified “first data” Is “process[ed] to determine information about” Either Transformation Pipeline.	16
1.	Petitioner Does Not Show That Barsness’s Monitoring Application Is “process[ed] to determine information about at least one of the first transformation pipeline and the second transformation pipeline” (Ground 1, All Claims).	17
2.	Petitioner’s Multiple-Reference Grounds Do Not Cure This Defect Of Its Single-Reference Barsness Ground (Grounds 2-4, All Claims).	23
C.	Even If The Distinct Barsness Items Invoked By Petitioner Could Satisfy Individual Limitations, Petitioner Does Not Show That They Are Part Of The Same “first data.”	24
III.	CONCLUSION	28

TABLE OF AUTHORITIES**Page(s)****COURT DECISIONS**

<i>Convolve, Inc. v. Compaq Computer Corp.</i> , 812 F.3d 1313 (Fed. Cir. 2016)	25
<i>In re Magnum Oil Tools Int’l, Ltd.</i> , 829 F.3d 1364 (Fed. Cir. 2016)	23
<i>Microsoft Corp. v. Biscotti, Inc.</i> , 878 F.3d 1052 (Fed. Cir. 2017)	9
<i>Microsoft Corp. v. FG SRC, LLC</i> , 860 F. App’x 708 (Fed. Cir. 2021) (nonprecedential).....	28
<i>Netflix, Inc. v. DivX, LLC</i> , 84 F.4th 1371 (Fed. Cir. 2023)	28
<i>PAR Pharm., Inc. v. TWI Pharms., Inc.</i> , 773 F.3d 1186 (Fed. Cir. 2014)	2
<i>Pers. Web Techs., LLC v. Apple, Inc.</i> , 917 F.3d 1376 (Fed. Cir. 2019)	9
<i>Persion Pharms. LLC v. Alvogen Malta Operations Ltd.</i> , 945 F.3d 1184 (Fed. Cir. 2019)	2
<i>Southwire Co. v. Cerro Wire LLC</i> , 870 F.3d 1306 (Fed. Cir. 2017)	9
<i>Trintec Indus., Inc. v. Top-U.S.A. Corp.</i> , 295 F.3d 1292 (Fed. Cir. 2002)	9

AGENCY DECISIONS

Xerox Corp. v. Bytemark, Inc.,
IPR2022-00624, Paper 9 (Aug. 24, 2022) (precedential).....22

STATUTES

35 U.S.C. § 312(a)(3).....27

PATENT OWNER'S EXHIBIT LIST

Exhibit	Description
2001	Declaration of Nathan Lowenstein in Support of Notice of Intent
2002	Declaration of Colette Woo in Support of Notice of Intent
2003	Declaration of Dennis J. Courtney in Support of Notice of Intent
2004	Defendant Palo Alto Networks, Inc.'s Preliminary Invalidity Contentions, <i>Qomplx LLC v. Palo Alto Networks, Inc.</i> , No. 2:25-cv-00913 (E.D. Tex.) (served Jan. 12, 2026) [Preliminary Invalidity Contentions]
2005	First Amended Docket Control Order, <i>Qomplx LLC v. Palo Alto Networks, Inc.</i> , No. 2:25-cv-00913 (E.D. Tex.) (ordered Jan. 3, 2026) [Docket Control Order]
2006	Time to Trial: Judge Rodney Gilstrap, DOCKET NAVIGATOR (last visited July 29, 2026) [Gilstrap Time to Trial]
2007	Defendant Microsoft Corporation's Opening Claim Construction Brief, <i>Qomplx LLC v. Microsoft Corporation</i> , No. 1:25-cv-01383 (W.D. Tex.) (served Mar. 23, 2026)
2008	Defendant Microsoft Corporation's Reply Claim Construction Brief, <i>Qomplx LLC v. Microsoft Corporation</i> , No. 1:25-cv-01383 (W.D. Tex.) (served Apr. 27, 2026)
2009	Plaintiff Qomplx LLC's Sur-Reply Claim Construction Brief, <i>Qomplx LLC v. Microsoft Corporation</i> , No. 1:25-cv-01383 (W.D. Tex.) (served May 11, 2026)
2010	Declaration and CV of Sam Malek, Ph.D. [Malek-Decl.]

All emphases in quotations are added unless otherwise stated.

I. INTRODUCTION

The Petition should be denied because it fails to establish a reasonable likelihood that any challenged claim is unpatentable. It never proves that the item it expressly identifies as the claimed “first data”—Barsness’s monitoring application 435—satisfies the linked requirements that claim 1 imposes on that same data. Claim 1 requires a memory of the third computer system to store “a first data”; requires that “the first data represents a portion of a distributed computational graph”; and later requires the third computer system to execute software instructions that “process the first data” to determine information about at least one of the claimed transformation pipelines. Ex. 1001 [’425] cl. 1. Petitioner expressly identifies monitoring application 435 as that “first data.” Pet., 39-40. Petitioner therefore had to show that the same identified “first data” satisfies the graph-representation and processing requirements. It does not.

The Petition repeatedly tries to fill those gaps through inherency. It identifies no express Barsness disclosure that Barsness’s monitoring application 435 includes the information on which Petitioner relies. Nor does Petitioner propose a modification that would create that relationship, much less explain why a POSITA would have been motivated to make such a modification with a reasonable expectation of success. Instead, Petitioner reasons that if Barsness’s monitoring application 435—or Barsness’s monitoring system 110, of which monitoring

application 435 is a part—can display, access, store, or process information, then that information must be included in monitoring application 435. But inherency may supply a missing limitation in an obviousness analysis only under a “high standard”: the limitation must be “necessarily present” or the “natural result” of the prior art’s express disclosures, and “probabilities or possibilities” are insufficient. *Persion Pharms. LLC v. Alvogen Malta Operations Ltd.*, 945 F.3d 1184, 1191 (Fed. Cir. 2019) (quoting *PAR Pharm., Inc. v. TWI Pharms., Inc.*, 773 F.3d 1186, 1195–96 (Fed. Cir. 2014)). The Petition does not even attempt to meet that standard.

For the graph-representation requirement, Ground 1 asserts that monitoring application 435 “includes” Barsness’s operator graph 132 because the application’s GUI can depict that graph. Pet., 40-43. But Barsness expressly stores operator graph 132 in a separate management system 105 and discloses that its monitoring system 110, which contains monitoring application 435, can communicate with management system 105. Ex. 1004 [Barsness] 6:43-51, 7:21-26, 10:12-20. The ability to depict remotely maintained graph information does not establish that operator graph 132—or other data representing it—is part of monitoring application 435. Petitioner’s expert assumes the missing relationship, Ex. 1003 [Min-Decl.] ¶¶ 102-103, and thereby conflates information available to an application with information “included” in it. *See* Section II.A *infra*.

The same inherency issue defeats Petitioner’s analysis of the claimed

processing requirement. Petitioner offers three alternatives—“Identifying Compute Nodes,” “Configuring Processing Elements,” and “Calculating Performance Metrics”—that respectively rely on processing operator graph 132, PE configurations, and tuple-processing data. Pet., 49-51. But none establishes that the relevant information is part of monitoring application 435. Barsness’s disclosure that monitoring system 110 can access, display, or store information in its memory 425 does not establish that the information is *part* of monitoring application 435. Again, Petitioner’s expert assumes the missing relationship, reasoning that Barsness’s GUI 510 “cannot display data it does not have.” Ex. 1003 [Min-Decl.] ¶ 108. But that reasoning does not establish the relationship required by Petitioner’s mapping. *See* Section II.B *infra*.

An additional, independent failure remains. Claim 1 does not permit Petitioner to use one “first data” for the graph-representation requirement and a different “first data” for the processing requirement: each reference to “*the* first data” refers back to the antecedent “*a* first data.” Even if the distinct Barsness items invoked by Petitioner could individually satisfy those limitations, Petitioner still had to establish that those items are constituents of the same “first data.” Instead, Petitioner begins with monitoring application 435, then gathers whichever separate Barsness disclosure resembles the next claim requirement, and declares the disclosed item “included” in the application without identifying any Barsness disclosure of

that relationship. The resulting mapping even requires Barsness's monitoring application 435 to serve as both the "first data" and the "third software instructions" that the claim requires to "process the first data," without explaining how Barsness discloses that relationship. Pet., 39-40, 48. In never presenting a consistent mapping of "the first data" across claim 1, Petitioner fails to show how this limitation is shown in its cited grounds. *See* Section II.C *infra*.

For each of these reasons, institution should be denied.

II. THE PETITION FAILS TO DEMONSTRATE A REASONABLE LIKELIHOOD OF SUCCESS.

A. Petitioner Does Not Show That Its Identified "first data" Represents A Portion Of A Distributed Computational Graph (All Grounds, All Claims).

Claim 1 requires a "third computer system" having a memory that stores "a first data," "wherein the first data represents a portion of a distributed computational graph." Pet., 3. Petitioner maps Barsness's monitoring system 110 to the "third computer system," memory 425 to the claimed memory, and monitoring application 435 to the "first data." Pet., 37-43; Ex. 1003 [Min-Decl.] ¶¶ 99-102. Petitioner must therefore show that monitoring application 435 represents a portion of a distributed computational graph.

Ground 1 asserts that monitoring application 435 "includes" Barsness's operator graph 132 (what Petitioner identifies as a "distributed computational graph") because the application's GUI can depict that graph. But Barsness places

operator graph 132 in a separate management system, and Petitioner identifies no express disclosure that the graph or other graph-representing information is included in monitoring application 435. *See* Section II.A.1 *infra*. Nor does Petitioner propose modifying Barsness to place graph-representing data in monitoring application 435; it presents that relationship as if Barsness already disclosed it. Ground 1 therefore depends on inherency. Petitioner posits that because Barsness’s GUI 510 can depict operator graph 132, Barsness’s monitoring application 435 necessarily contains the graph or other graph-representing data. But neither the Petition nor Petitioner’s expert attempts to establish that necessity. Grounds 2-4 do not cure that failure: none changes Ground 1’s identification of monitoring application 435 as the “first data” or shows that the application represents the claimed graph portion. *See* Section II.A.2 *infra*.

1. Petitioner Does Not Show That Barsness’s Monitoring Application “represents a portion of a distributed computational graph” (Ground 1, All Claims).

Claim 1 requires that “the first data represents a portion of a distributed computational graph.” Petitioner maps Barsness’s operator graph 132 to the “distributed computational graph.” Pet., 43-46. Because Petitioner maps monitoring application 435 to the “first data,” it must show that the application “represents a portion of” operator graph 132. Petitioner attempts to make that showing by asserting that monitoring application 435 “includes” operator graph 132. Barsness

does not disclose that relationship. Instead, Barsness expressly locates operator graph 132 elsewhere.

Petitioner contends that “monitoring application 435 (first data) includes operator graph 132.” Pet., 40-41. But the cited support shows only that application 435 includes GUI 510, which Barsness says may be “configured to depict the operator graph 132.” Pet., 40-41 (citing Ex. 1004 [Barsness] 3:45-50, 11:53-61; Ex. 1003 [Min-Decl.] ¶¶ 102-103). A disclosure that an application’s GUI can depict a graph is not a disclosure that the application “includes” the graph or other data representing it.

Barsness instead expressly places operator graph 132 in management system 105. It does not disclose that operator graph 132—or any other data representing a portion of that graph—is part of monitoring application 435. Barsness’s overview states that “the management system 105 includes an operator graph 132 and a stream manager 134.” Ex. 1004 [Barsness] 6:43-53. Figure 1A likewise depicts operator graph 132 inside management system 105. That same figure identifies monitoring system 110—which includes monitoring application 435—as a separate networked component:

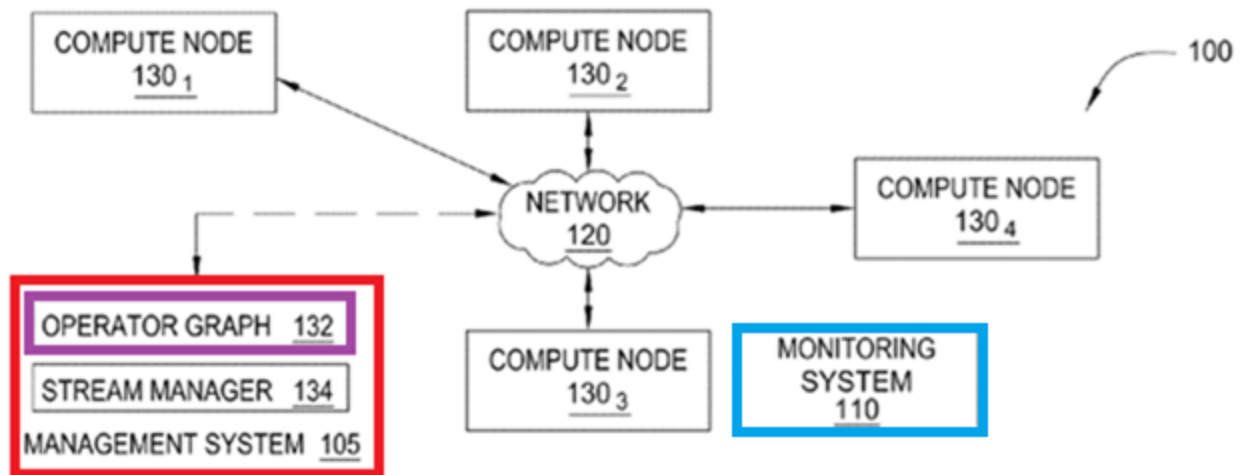


FIG. 1A

Ex. 1004 [Barsness] Fig. 1A.

Barsness later states that **management system 105's** (*not* **monitoring system 110's**) “storage 330 also includes the **operator graph 132**,” and that “**processing element and operator source code 345** may also be stored in the storage 330.” Ex. 1004 [Barsness] 10:12-20, 10:48-56. Figure 3 depicts those items within **management system 105**:

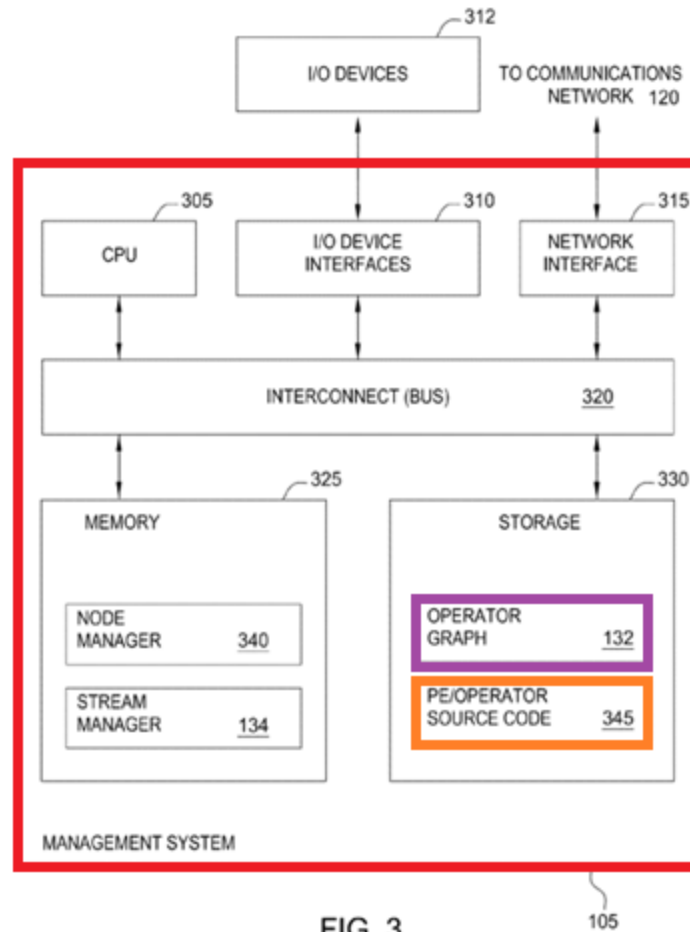


FIG. 3

Ex. 1004 [Barsness] Fig. 3.

Like Barsness, the Petition too treats management system 105 and monitoring system 110 as distinct computer systems. *See, e.g.*, Pet., 23-25; Ex. 1004 [Barsness] Fig. 1A, 6:43-53, 7:15-26. Petitioner cites no disclosure placing operator graph 132—or other data representing a graph portion—within monitoring application 435. *See* Ex. 1004 [Barsness] Figs. 3-4, 10:12-20, 11:7-37.

GUI 510's ability to depict the graph does not bridge that gap. Barsness says only that GUI 510 *may* depict the graph and related information. Ex. 1004

[Barsness] 11:53-12:12. That disclosure does not expressly show that operator graph 132 or related information is *included* in monitoring application 435. Nor does Petitioner propose modifying Barsness to create that relationship, much less explain why a POSITA would have been motivated or able to make such a modification.

This is an inherency theory. *See Microsoft Corp. v. Biscotti, Inc.*, 878 F.3d 1052, 1065 (Fed. Cir. 2017). But inherency requires that a feature be “necessarily present, not merely probably or possibly present, in the prior art.” *See, e.g., Trintec Indus., Inc. v. Top-U.S.A. Corp.*, 295 F.3d 1292, 1295 (Fed. Cir. 2002) (cleaned up); *see also, e.g., Pers. Web Techs., LLC v. Apple, Inc.*, 917 F.3d 1376, 1382–83 (Fed. Cir. 2019) (reversing IPR obviousness determination because a “proposed, theoretical” data structure did not necessarily exist in the reference; “mere possibility is not enough”); *Southwire Co. v. Cerro Wire LLC*, 870 F.3d 1306, 1310–11 (Fed. Cir. 2017) (Board erred by relying on inherency without a finding that a missing limitation “necessarily would” result from a prior art process). Neither the Petition nor Petitioner’s expert explains why the ability of Barsness’s GUI 510 to depict information about operator graph 132 would necessarily require the operator graph 132—or other data representing it—to be contained in Barsness’s monitoring application 435.

Indeed, Barsness’s system provides for the cited display functionality without requiring Petitioner’s asserted inclusion. Barsness’s monitoring system 110

communicates with management system 105, where operator graph 132 is stored. Ex. 1004 [Barsness] 7:21-26, 10:48-56. As Dr. Malek explains, this communication path permits GUI 510 to obtain information for display without requiring operator graph 132 or other graph-representing information to be stored within Barsness's monitoring application 435:

A POSITA would not understand Barsness's disclosure that its GUI 510 is "configured to depict the operator graph 132," Ex. 1004 [Barsness] 11:53-61, to require that monitoring application 435 include operator graph 132 (or other graph-representing information) within itself. A graphical user interface can render information obtained from another software component or data store without maintaining a separate copy of the underlying data within the GUI or the application that presents it. Indeed, it is *typical* for a graphical user interface to depict data that it "does not have" literally stored within itself. *Cf.* Ex. 1003 [Min-Decl.] ¶ 108.

Ex. 2010 [Malek-Decl.] ¶ 41. Indeed, when one opens and views a document in Adobe Acrobat, Microsoft Word, or a web browser, for example, though a graphical user interface within the application may visually present the document, the document remains only an input document accessed by the application from a separate location; it is not thereby incorporated into the application's own executable code, application definition, or constituent components—not persistently, and not even temporarily, during user interaction with the document via the graphical user

interface. *Id.*, ¶ 42. As Dr. Malek further explains:

Barsness expressly locates operator graph 132 in storage 330 of management system 105, which Barsness depicts as a separate component from monitoring system 110. Ex. 1004 [Barsness] Figs. 1A, 3, 6:43-51, 10:12-19, 10:48-56. Barsness also discloses that monitoring system 110 can communicate with its management system 105. *Id.*, 7:21-26. Even assuming that Barsness's GUI 510 stores some information that it displays somewhere within its monitoring system 110, this would not establish that such information is stored as part of monitoring application 435.

Ex. 2010 [Malek-Decl.] ¶ 43. Indeed, dynamically created application data is ordinarily stored separately from an application's executable implementation so that it can change and persist independently:

Application data can be distinguished based on whether it is fixed within an application's implementation or generated during operation. Some application data, such as hard-coded configuration values, default templates, static resources, lookup tables, or other information packaged with the source code or compiled artifact, may be embedded in an application itself. This data is typically static: it exists before the application runs, and it changes only when the application is modified, rebuilt, or redeployed. Dynamically created application data, by contrast—such as user inputs, transaction records, session data, logs, uploaded files, generated outputs, or state changes—is typically produced, received, modified, or accumulated while an application is running. This runtime data is typically stored outside the application

implementation, for example in a database, file system, cache, object store, or external service, so it can persist independently of the application code and be accessed or updated across executions.

Ex. 2010 [Malek-Decl.] ¶ 44.

Barsness's operator graph 132 is dynamically changing application data of that kind. Barsness states that processing elements may be "inserted or removed dynamically" from its operator graph, that monitoring application 435 may identify or make modifications to the running stream application, and that stream manager 134 may "change the structure of the operator graph 132" by moving, inserting, removing, or otherwise modifying processing elements. Ex. 1004 [Barsness] 3:28-45, 7:27-37. As Dr. Malek explains, those disclosures would lead a POSITA to expect operator graph 132 to be maintained separately from the executable implementation of monitoring application 435, rather than embedded within it. Ex. 2010 [Malek-Decl.] ¶ 45. Such a design would be consistent with Barsness's express disclosures without requiring Petitioner's asserted inclusion. It therefore defeats Petitioner's claim of inherency.

Petitioner therefore never establishes the inclusion relationship on which its mapping depends. It identifies monitoring application 435 as the claimed "first data" and asserts that the application "includes" operator graph 132 or data representing it because GUI 510 can depict the graph. But Barsness expressly locates operator graph 132 in its separate management system 105 and provides a communication

path through which monitoring application 435 can obtain information about the graph. Petitioner identifies no Barsness disclosure that operator graph 132 or other graph-representing information is included in monitoring application 435. Petitioner neither proves that relationship as inherent nor proposes a modification that would create it. Ground 1 therefore fails to show that the identified “first data” “represents a portion of a distributed computational graph.”

2. Petitioner’s Multiple-Reference Grounds Do Not Cure This Defect Of Its Single-Reference Barsness Ground (Grounds 2-4, All Claims).

Petitioner’s multiple-reference grounds likewise fail to show the claimed graph-representing “first data” or cure Ground 1’s failed inherency theory. Claim 1 requires the third computer system to comprise a memory storing “a first data,” and requires that “the first data represents a portion of a distributed computational graph.” Under Petitioner’s mapping, Barsness’s monitoring system 110 is the “third computer system” and monitoring application 435 is the “first data.” Pet., 37-43. Nothing in Grounds 2-4 changes that identification, supplies the missing inclusion relationship, or otherwise shows that Barsness’s monitoring application 435 represents a portion of Barsness’s operator graph 132, which Petitioner identifies as a “distributed computational graph.”

In particular, Petitioner’s Ground 2 does not cure Ground 1’s failure. Ground 2 expressly incorporates Ground 1’s claim 1 analysis and does not separately

address limitation 1[d] (“wherein a third computer system of the second plurality of computer systems comprises a memory that stores a first data”), which introduces “a first data.” *Id.*, 74-75.¹ It therefore leaves unchanged Ground 1’s identification of monitoring application 435 as the claimed “first data.” Ground 2 separately addresses limitation 1[d][i] (“wherein the first data represents a portion of a distributed computational graph”), but only by showing that Chakradhar discloses a graph or topology. *Id.*, 75. It never shows that the antecedent “first data”—monitoring application 435—represents that graph, or otherwise cures Ground 1’s failure to show that monitoring application 435 “represents a portion of a distributed computational graph.”

Ground 2 does purport to add Chakradhar’s “route-maps” to a combination with Barsness, but Petitioner places these “route-maps” with operators at Barsness’s compute nodes. *Id.*, 72, 74 (citing Ex. 1005 [Chakradhar] [0122]; Ex. 1003 [Min-Decl.] ¶¶ 162, 165). Petitioner does not place Chakradhar’s route-maps in Barsness’s monitoring system 110, make them part of Barsness’s monitoring application 435, or explain why a POSITA would make either modification.

Petitioner does state elsewhere, in its general combination discussion, that

¹ For ease of reference, this brief deploys the Petition’s labels for various elements of claim 1. Pet., 3-4.

applying Chakradhar's route-maps would allow monitoring system 110 to store unspecified "respective data" representing a portion of a distributed computational graph. *Id.*, 73. But that sentence does not identify the "respective data," map it to the "first data" of limitation 1[d], or alter Ground 1's express mapping of Barsness's monitoring application 435 as that "first data." Ground 2's limitation-specific analysis likewise supplies none of those missing connections. Ground 2 therefore does not cure Ground 1's failure to show that the identified "first data"—monitoring application 435—"represents a portion of a distributed computational graph."

Petitioner's Barsness-Siripurapu Ground 3 likewise does not cure Ground 1's failure. Ground 3 incorporates Ground 1 and separately addresses only limitations 1[e] through 1[h], not limitations 1[d] or 1[d][i]. *Id.*, 82-83. Although Ground 3 later labels Siripurapu's worker-node status and processing-load metadata as "first data" in addressing limitation 1[e][i] ("process the first data to determine information about at least one of the first transformation pipeline and the second transformation pipeline"), neither Petitioner nor Petitioner's expert argues that this metadata "represents a portion of a distributed computational graph," as limitation 1[d][i] requires. *Id.*, 83; Ex. 1003 [Min-Dec1.] ¶ 179.

Petitioner's Ground 4 adds no separate limitation-specific analysis; it only purports to combine the Chakradhar and Siripurapu theories "for the same reasons" offered in Grounds 2 and 3. *Pet.*, 84. Ground 4 therefore fails for the same reasons.

B. Petitioner Does Not Show That Its Identified “first data” Is “process[ed] to determine information about” Either Transformation Pipeline.

Claim 1 separately requires the third computer system to execute software instructions that “process the first data to determine information about at least one of the first transformation pipeline and the second transformation pipeline.” Petitioner identifies monitoring application 435 as the “first data.” Pet., 39-40. As explained above, Petitioner already fails to show that this alleged “first data” represents a portion of a distributed computational graph, as claim 1 requires. *See* Section II.A *supra*.

The processing limitation imposes a second requirement: Petitioner’s alleged “third software instructions” must *process* that same “first data.” Ground 1 offers three alternatives—“Identifying Compute Nodes,” “Configuring Processing Elements,” and “Calculating Performance Metrics”—that respectively rely on processing operator graph 132, PE configurations, and tuple-processing data. Pet., 49-51. Each alternative depends on the same inherency inference: information that Barsness’s monitoring application 435 or Barsness’s monitoring system 110 can access, display, store, or process must be included in Barsness’s monitoring application 435 itself. But Petitioner identifies no Barsness disclosure of any such inclusion and proposes no modification to create it. Even assuming, *arguendo*, that monitoring application 435—what Petitioner maps to the “third software

instructions”—is generally capable of processing the identified categories of information, Petitioner still must show that the information is part of monitoring application 435—the alleged “first data.” But none of Petitioner’s three theories does so. *See* Section II.B.1 *infra*.

Petitioner’s Grounds 2-4 do not cure that failure. Ground 2 does not separately address the processing limitation at all. Ground 3 inconsistently labels different Siripurapu metadata as “first data,” without showing that the metadata is part of the “first data” identified in Ground 1. Ground 4 adds no separate analysis. *See* Section II.B.2 *infra*. Petitioner therefore fails across all grounds to show that “the first data” is “process[ed]” as the claims require.

1. Petitioner Does Not Show That Barsness’s Monitoring Application Is “process[ed] to determine information about at least one of the first transformation pipeline and the second transformation pipeline” (Ground 1, All Claims).

Claim 1 requires the third computer system to execute software instructions that “process the first data to determine information about at least one of the first transformation pipeline and the second transformation pipeline.” Petitioner identifies monitoring application 435 as the “first data.” Pet., 39-40. Yet its three theories for this limitation rely on processing different information: operator graph 132, PE configurations, and tuple-processing data. *Id.*, 49-51. Because Petitioner identifies no express Barsness disclosure that any of the cited information is part of Barsness’s monitoring application 435 and proposes no modification to make it so,

each theory again depends on inherency. Each theory fails because the asserted inclusion does not necessarily follow from Barsness's disclosure. *See* Section II.A.1 *supra*.

Petitioner's first theory, "Identifying Compute Nodes," depends on its assertion that operator graph 132 is part of monitoring application 435. As explained above, however, Petitioner never shows that operator graph 132 is part of monitoring application 435. *See* Section II.A.1 *supra*. Accordingly, even assuming that Barsness's monitoring application 435 processes Barsness's operator graph 132, Petitioner has not shown that the object being processed is Barsness's monitoring application 435 itself—which is what Petitioner identified as the claimed "first data."

Petitioner's "Configuring Processing Elements" theory similarly fails because it does not show that the cited PE-configuration information is part of monitoring application 435. Petitioner attempts to make that connection by simply labeling a "'configuration' of PEs" as "***monitoring application 435's*** 'configuration' of PEs." Pet., 50. But Barsness does not disclose that the PE configurations are stored in monitoring application 435. Barsness says that a user may configure monitoring system 110 to specify which data to collect, that monitoring system 110 may configure compute nodes to collect that data "accordingly," and that GUI 510 may display aspects of PE configurations. Ex. 1004 [Barsness] 12:3-12, 13:49-55. Petitioner's expert then infers from GUI 510's display capability that the PE

configurations are stored in memory 425 as part of monitoring application 435, reasoning that GUI 510 “cannot display data it does not have.” Ex. 1003 [Min-Decl.] ¶ 108. But this inference merely repeats the failed theory addressed above: that an application’s ability to access or display information necessarily means that the application contains that information. *See* Section II.A.1 *supra*.

That is another inherency theory, and it fails for the same reason. Even if Petitioner’s PE “configurations” are stored somewhere in memory 425 of Barsness’s monitoring system 110, it does not necessarily follow that they are part of monitoring application 435. Petitioner identifies no Barsness disclosure establishing the asserted relationship, and Petitioner’s expert does not explain why that relationship is necessary. As Dr. Malek explains:

As I have observed above, application data that is generated or changed while a program is running is ordinarily maintained separately from a program’s executable implementation. Barsness’s PE configurations are that kind of runtime data: Barsness discloses assigning and reassigning processing elements among compute nodes, configuring the data attributes that processing elements process or collect, and reconfiguring the stream application while it runs. *See, e.g.*, Ex. 1004 [Barsness] 10:14-39, 11:26-37, 13:49-55. A POSITA would understand that it is possible for these configurations to be capable of changing without changes to the monitoring application 435 itself. In particular, a POSITA would not infer from GUI 510’s ability to display

configuration information that the configurations are embedded in, or stored as part of, monitoring application 435.

Ex. 2010 [Malek-Decl.] ¶ 56.

Petitioner's "Calculating Performance Metrics" theory likewise relies on information not shown to be part of monitoring application 435. Petitioner says monitoring system 110 "process[es] monitoring application 435's 'tuple processing data'" to calculate performance metrics. Pet., 50-51. But merely labeling Barsness's "tuple processing data" as "*monitoring application 435's* 'tuple processing data'" does not establish that the data is part of Barsness's monitoring application 435. Barsness discloses that its compute nodes transfer collected tuple data to monitoring system 110; that monitoring system 110 "persistently stores the collected data"; that the collected data may be stored "in a database or in a file"; and that monitoring system 110 may cache the data in memory 425. Ex. 1004 [Barsness] 13:32-45. Barsness further states that "[t]he transformed collected data is stored either in a database or in the memory 425." Ex. 1004 [Barsness] 13:56-67. Such disclosures show at most that monitoring system 110 can store tuple-processing data. They do not show that it stores this data as *part of* Barsness's monitoring application 435.

Barsness's Figure 4 reinforces the distinction. It separately depicts "monitoring application 435" in memory 425 and "historical data 440" in storage 430:

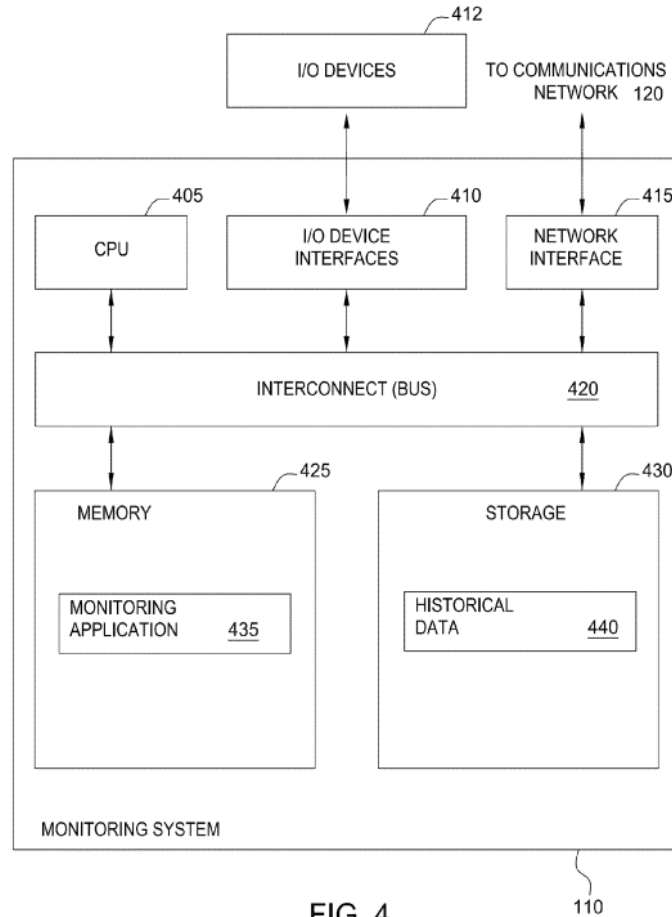


FIG. 4

Ex. 1004 [Barsness] Fig. 4.

Barsness never states or suggests that data created and stored during operation of monitoring system 110—even if stored in the same memory 425 that contains monitoring application 435—thereby becomes *part of* monitoring application 435. Petitioner’s expert again reasons that because tuple-processing data and performance metrics may be stored in memory 425 and presented through GUI 510, they are “part of the monitoring application.” Ex. 1003 [Min-Decl.] ¶¶ 104-107. But storage in the same memory, like access or display, does not establish that the information is part of the application. Petitioner identifies no Barsness disclosure making tuple-

processing data part of monitoring application 435 merely because both may reside in memory 425. Accordingly, even assuming that Barsness's monitoring application 435 processes the tuple-processing data discussed under the "Calculating Performance Metrics" theory, Petitioner has not shown that processing that data constitutes processing the identified "first data."

All three processing theories thus fail for the same reason. Petitioner identifies monitoring application 435 as the "first data," but then relies on processing operator graph 132, PE configurations, or tuple-processing data without establishing that any of those items is part of monitoring application 435. The asserted inclusion relationships rest on Petitioner's expert's inferences from display, access, or storage, not on an express disclosure. *See Id.*, ¶¶ 102-110. Those inferences do not establish the necessity that inherency requires.

Nor does Petitioner's expert provide facts or technical reasoning supporting the asserted inclusion relationships. Under 37 C.F.R. § 42.65(a), however, an expert opinion that does not disclose its underlying facts or data is entitled to little or no weight. And testimony that simply repeats an otherwise unsupported conclusion without additional evidence or technical reasoning likewise "adds little" to that conclusion. *Xerox Corp. v. Bytemark, Inc.*, IPR2022-00624, Paper 9, 15-16 (Aug. 24, 2022) (precedential); *see also In re Magnum Oil Tools Int'l, Ltd.*, 829 F.3d 1364, 1380 (Fed. Cir. 2016) (obviousness cannot be established through "mere conclusory

statements”).

Ground 1 therefore fails to show that “the first data” is “process[ed]” as claim 1 requires.

2. Petitioner’s Multiple-Reference Grounds Do Not Cure This Defect Of Its Single-Reference Barsness Ground (Grounds 2-4, All Claims).

Petitioner’s Ground 2 does not cure Ground 1’s failure. It expressly incorporates Ground 1’s claim 1 analysis and separately addresses only limitations 1[a] and 1[d][i]. Pet., 74-76. Ground 2 never separately addresses limitation 1[e][i], which requires the “third software instructions” to “process the first data to determine information about at least one of the first transformation pipeline and the second transformation pipeline.” It therefore inherits Ground 1’s failure to show that “the first data” is processed as required.

Ground 3 likewise does not cure the defect. Ground 3 incorporates Ground 1 and does not separately address limitation 1[d], where Petitioner identifies monitoring application 435 as the “first data.” *Id.*, 82-83. Yet when Ground 3 reaches limitation 1[e][i], Petitioner simply labels different Siripurapu worker-node metadata—including “current status” and “current processing load”—as “first data.” *Id.*, 83 (citing Ex. 1003 [Min-Decl.] ¶ 179). Merely placing the label “(first data)” next to newly introduced Siripurapu information does not establish that this information is part of the “first data” Petitioner identified in Ground 1. Neither

Petitioner nor Petitioner's expert explains how Siripurapu's metadata becomes part of Barsness's monitoring application 435. Ground 3 therefore does not show that processing the Siripurapu metadata constitutes processing "the first data" as limitation 1[e][i] requires.

Ground 4 adds no separate analysis of limitation 1[e][i]. It merely combines the Chakradhar and Siripurapu theories "[f]or the same reasons" offered in Grounds 2 and 3. *Id.*, 84. Ground 4 therefore does not cure the defect either.

Thus, Grounds 2-4 all fail to show that Petitioner's alleged "first data" is "process[ed] to determine information about at least one of the first transformation pipeline and the second transformation pipeline."

C. Even If The Distinct Barsness Items Invoked By Petitioner Could Satisfy Individual Limitations, Petitioner Does Not Show That They Are Part Of The Same "first data."

Sections II.A and II.B show that Petitioner fails to establish two separate properties of what it identifies as the "first data." Those failures share the same defect: Petitioner repeatedly treats inclusion relationships as inherent without proving necessity. A further defect remains even if those failures are set aside: claim 1 repeatedly refers to the *same* antecedent "first data." Petitioner therefore had to identify one item as the "first data" and use that same item in both its graph-representation and processing analyses. *See, e.g., Convolve, Inc. v. Compaq Computer Corp.*, 812 F.3d 1313, 1321 (Fed. Cir. 2016) (explaining that "the

processor” referred back to an earlier-recited “a processor”). Petitioner never does so. For limitation 1[d], it expressly identifies monitoring application 435 as the “first data.” Pet., 39-40. Its later analyses invoke disclosures relating to Barsness’s operator graph 132, PE configurations, and tuple-processing data. *Id.*, 40-43, 49-51. But Petitioner never shows that these items are part of Barsness’s monitoring application 435. *See* Section II.B *supra*. Thus, even assuming that individual Barsness items could separately satisfy isolated claim requirements, the Petition never shows that the same “first data” satisfies both limitations. This approach is claim-driven rather than reference-driven: Petitioner uses the claim as a checklist, gathers whichever Barsness disclosure resembles the next claim requirement, and declares the disclosed item to be part of monitoring application 435. But Petitioner must prove the relationships required by the claim from Barsness’s disclosure; it cannot create those relationships simply by labeling separately disclosed information as part of its identified “first data.”

Petitioner’s inherency logic also has no limiting principle. Barsness says that the GUI 510 of its monitoring system may depict compute nodes, PEs, and operators; permit a user to view PE logs and configurations; and display performance information. Ex. 1004 [Barsness] 11:53-12:12. Barsness also says that monitoring application 435 “interact[s] with” data collectors at compute nodes to retrieve PE- and operator-related information. *Id.*, 12:13-21. Barsness locates the PEs,

executable operators, and data collector 250 at compute nodes—not in monitoring application 435. *See, e.g., id.*, Fig. 2, 8:66–9:33. If the ability to access or display information necessarily made the underlying data part of Barsness’s monitoring application 435, the same reasoning would also sweep in the PE-, operator-, and data-collector information that Barsness maintains at its compute nodes. Petitioner provides no basis for drawing the line where its claim mapping requires it.

Petitioner’s mapping also produces another unexplained relationship: it requires Barsness’s monitoring application 435 to process itself. Petitioner maps monitoring application 435 both to the “first data” and to the “third software instructions” that “process the first data.” Pet., 39-40, 48; Ex. 1003 [Min-Decl.] ¶¶ 101, 121. Patent Owner does not contend that an application can never process itself or store within itself some of the data on which it relies. But Petitioner never explains why Barsness’s monitoring application 435 does so. As Dr. Malek explains:

Barsness generally assigns distinct roles to executable instructions and the data processed by those instructions. Barsness describes its CPU 405 as retrieving and executing its monitoring application 435, Ex. 1004 [Barsness] 11:7-11, 11:26-30, while Barsness separately discloses, in one example, that a “data collector module 515” retrieves PE data and stores it as historical data 440 (depicted separately from its monitoring application, *see id.*, Fig. 4), while in another example, a “data analysis module 530” compares current tuple rates

with that historical data. *See id.*, 11:38-49, 12:13-30. Barsness never describes its monitoring application as using *itself* as an input, and never identifies any part of its monitoring application as storing the data that it operates on.

That distinction matters, for example, to limitation 1[e][i]. Petitioner maps monitoring application 435 to the “third software instructions,” but previously mapped the same application to the “first data” those instructions must “process.” Ex. 1003 [Min-Decl.] ¶¶ 101, 121-125. Absent some disclosure in Barsness of the monitoring application processing *itself* to determine other information about the Barsness system, a POSITA would not understand Barsness to disclose the relationship required by Petitioner’s mapping.

Ex. 2010 [Malek-Decl.] ¶¶ 69-71.

The Board should not construct a coherent “first data” theory on Petitioner’s behalf. An IPR petition must identify “with particularity” the evidence supporting each ground. 35 U.S.C. § 312(a)(3). “[A] petitioner may not rely on a vague, generic, and/or meandering petition and later fault the Board for failing to understand what the petition really meant”; “[u]ltimately, it is the petitioner’s burden to present a clear argument.” *Netflix, Inc. v. DivX, LLC*, 84 F.4th 1371, 1377 (Fed. Cir. 2023). Nor is it the Board’s task “to cobble together assertions from different sections of a petition or citations of various exhibits in order to infer every possible permutation of a petitioner’s arguments.” *Microsoft Corp. v. FG SRC, LLC*, 860 F. App’x 708, 713 (Fed. Cir. 2021) (nonprecedential).

Here, Petitioner never presents a single coherent mapping of “the first data” across claim 1. This failure to identify how “the first data” is present in its cited grounds is independently fatal.

III. CONCLUSION

For the foregoing reasons, institution should be denied.

Respectfully submitted,

Dated: September 2, 2026

By: / Kenneth J. Weatherwax /
Kenneth J. Weatherwax (Reg. No. 54,528)
Nathan Lowenstein (*pro hac vice*)
Dennis Courtney (*pro hac vice*)
Colette Woo (*pro hac vice*)
LOWENSTEIN & WEATHERWAX LLP

CERTIFICATE OF COMPLIANCE WITH TYPE-VOLUME LIMITS

This Patent Owner's Preliminary Response ("POPR") consists of 5,604 words, excluding table of contents, table of authorities, certificate of service, this certificate, or table of exhibits. The POPR complies with the type-volume limitation of 14,000 words as mandated in 37 C.F.R. § 42.24. In preparing this certificate, counsel has relied on the word count of the word-processing system used to prepare the paper (Microsoft Word).

Respectfully submitted,

Dated: September 2, 2026

By: / Michael Cosley /
Michael Cosley

CERTIFICATE OF SERVICE

The undersigned hereby certifies that the following documents were served by electronic service, by agreement between the parties, on the date below:

PATENT OWNER'S PRELIMINARY RESPONSE

EXHIBIT 2010

The names and email addresses of the parties being served are as follows:

James M. Glass	jimglass@quinnemanuel.com
Quincy Lu	quincylu@quinnemanuel.com
Brian Biddinger	brianbiddinger@quinnemanuel.com
Jeremy Thomas	jeremythomas@quinnemanuel.com
	qe-pan-qomplx-ipr@quinnemanuel.com

Respectfully submitted,

Dated: September 2, 2026

By: / Michael Cosley /
Michael Cosley