

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

MICROSOFT CORPORATION,
Petitioner,

v.

QOMPLX LLC,
Patent Owner.

Case IPR2026-00298
Patent 12,143,424

PATENT OWNER'S PRELIMINARY RESPONSE

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
II.	THE PETITION FAILS TO DEMONSTRATE A REASONABLE LIKELIHOOD OF SUCCESS.....	3
A.	Petitioner Fails To Show A Plurality of Computer Systems, Wherein “each” System “stores a respective first data, wherein the respective first data represents a respective portion of a distributed computational graph” (Grounds 1 & 3, Claims 1-3, 5, 7).	3
1.	Petitioner Does Not Consistently Identify The Claimed “respective first data.”	6
2.	Petitioner Fails To Show That Barsness’s Compute Nodes Store “respective first data [that] represents a respective portion of a distributed computational graph.”.....	9
3.	Petitioner’s Arguments Regarding Dependent Claims And The Barsness-Siripurapu Combination Do Not Cure These Defects.	13
B.	Petitioner Fails To Show That Barsness Teaches “first data that ... identifies at least a second transformation pipeline” (Grounds 1 & 3, Claims 9-11, 13).....	14
1.	Petitioner’s Argument By Reference Fails.	15
2.	Petitioner’s Argument That Barsness’s Compute Node Is A “second transformation pipeline” Does Not Show How Petitioner’s Own Construction Is Satisfied.	17
3.	Petitioner’s Arguments Regarding Dependent Claims And The Barsness-Siripurapu Combination Do Not Cure These Defects.	20
C.	Petitioner Fails To Demonstrate That Barsness Teaches The “third computer system of the first plurality” Limitations (All Grounds, All Challenged Claims).....	21

1.	Petitioner’s Ground 1 Argument Fails.....	21
a)	The Activity Petitioner Identifies In Barsness’s Management System Is Initiated By Barsness’s Monitoring System.	23
b)	Petitioner Does Not Show That Barsness’s Management System Executes Software Instructions That Cause A Fourth Computer System To Execute Software Instructions Applying The First Or Second Transformation Pipeline.	26
2.	Petitioner’s Grounds 2-4 Arguments Also Fail.....	29
D.	Petitioner Does Not Show A Motivation To Combine Barsness With Chakradhar With A Reasonable Expectation Of Success (Grounds 2, 4, All Challenged Claims).....	31
1.	Chakradhar’s “Route-Maps” Are Created And Updated By Chakradhar’s Controller.....	34
2.	Petitioner Does Not Show A Reasonable Expectation Of Success Because It Does Not Explain How Route-Maps Are Created Or Updated In Its Combination.....	38
E.	Petitioner Does Not Show A Motivation To Combine Barsness With Siripurapu Or A Reasonable Expectation of Success (Grounds 3 & 4, All Challenged Claims).....	41
III.	CONCLUSION	44

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

<i>ActiveVideo Networks, Inc. v. Verizon Commc 'ns, Inc.</i> , 694 F.3d 1312 (Fed. Cir. 2012)	44
<i>Apple Inc. v. Cellular Comm 'ns Equipment, LLC</i> , IPR2015-00576, Paper 7 (June 12, 2015).....	43
<i>Convolve, Inc. v. Compaq Computer Corp.</i> , 812 F.3d 1313 (Fed. Cir. 2016)	10
<i>Hulu LLC v. Sound View Innovations LLC</i> , IPR2018-00582, Paper 34 (Aug. 5, 2019) (informative).....	43
<i>In re Schweickert</i> , 676 F. App'x 988 (Fed. Cir. 2017) (nonprecedential).....	43
<i>Intelligent Bio-Systems, Inc. v. Illumina Cambridge, Ltd.</i> , 821 F.3d 1359 (Fed. Cir. 2016)	33
<i>Medtronic, Inc. v. Teleflex Innovations S.A.R.L.</i> , 69 F.4th 1341 (Fed. Cir. 2023)	33
<i>Microsoft Corp. v. FG SRC, LLC</i> , 860 F. App'x 708 (Fed. Cir. 2021) (nonprecedential).....	18
<i>Netflix, Inc. v. DivX, LLC</i> , 84 F.4th 1371 (Fed. Cir. 2023)	16, 18
<i>PersonalWeb Techs., LLC v. Apple, Inc.</i> , 848 F.3d 987 (Fed. Cir. 2017)	40, 44
<i>R.J. Reynolds Vapor Co. v. Fontem Holdings I B.V.</i> , IPR2016-01692, Paper 45 (Mar. 2, 2018)	43
<i>Virtek Vision Int'l ULC v. Assembly Guidance Sys.</i> , 97 F.4th 882 (Fed. Cir. 2024)	43

STATUTES

35 U.S.C. § 312(a)(3).....16

PATENT OWNER'S EXHIBIT LIST

Exhibit	Description
2001	Declaration and CV of Dr. Sam Malek

All emphases in quotations are added unless otherwise stated.

I. INTRODUCTION

Institution should be denied because the Petition fails to show a reasonable likelihood of prevailing on any challenged claim.

The '424 patent's claims require more than the generic, strawman distributed computing system that the Petition purports to analyze. Independent claim 1 and its dependent claims, for example, require each computer system in a claimed first plurality to store "first data" representing a respective portion of a "distributed computational graph." Independent claim 9 and its dependent claims require each system in a first plurality to store "first data" identifying "at least a second transformation pipeline." Petitioner points to a graph in Barsness—and even to various functions disclosed in Barsness—but when applying the claims, disregards the actual structure of the claimed distributed computing clusters, and the manner in which the claimed clusters actually leverage the claimed distributed computational graph and data about the multiple claimed transformation pipelines. Petitioner's grounds that combine Barsness with other references do not cure these failures.

As one example, the Petition first advances a single-reference Barsness ground. Petitioner identifies a supposed plurality of computer systems that includes Barsness's "management system" and "compute nodes," but does not show that each of these elements stores first data that represents a respective portion of a "distributed computational graph" (claim 1 and its dependent claims) or identifies "at least a

second transformation pipeline” (claim 9 and its dependent claims). In fact, Barsness does not show how Barsness’s compute nodes represent any part of the graph that Petitioner associates with Barsness’s management system, and does not explain how Barsness teaches “a second transformation pipeline.” See Sections II.A, II.B, *infra*.

As another example, the Petition also fails to show any causation, whereby one computer system in a first plurality causes another computer to execute software instructions applying a “transformation pipeline.” See Section II.C, *infra*. Because Petitioner has mapped the claimed computer system to Barsness’s “management system,” Petitioner strains to give Barsness’s management system this role, yet fails to identify how it *causes* other systems to apply what Petitioner identifies as a “transformation pipeline.” See *id*.

The asserted Chakradhar combinations do not cure these defects. Petitioner’s alleged motivation to combine depends on the supposed benefits of using Chakradhar’s dynamically updated “route-maps,” but the Petition imports only the “route-map” data structures from Chakradhar into Barsness, while omitting the component that creates and updates them. Petitioner therefore does not explain how the combined system would obtain, update, or use the route-maps on which the proposed benefits depend. See Section II.D, *infra*.

Petitioner’s Siripurapu combinations fare no better. Petitioner invokes

Siripurapu for its supposed “scalability teachings,” Pet., 80, but simultaneously contends that Barsness already provides the very scaling that Siripurapu is said to supply. And Petitioner never explains how Siripurapu’s “conductor”—a component used in Siripurapu for load balancing, and a component that Petitioner analogizes to Barsness’s management system, *see* Pet., 81—would be integrated with Barsness’s management system in Petitioner’s proposed combination. *See* Section II.E, *infra*.

Petitioner has failed to carry its burden to make out even a *prima facie* case of unpatentability—let alone to show a reasonable likelihood of unpatentability. Institution should therefore be denied.

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

A. Petitioner Fails To Show A Plurality of Computer Systems, Wherein “each” System “stores a respective first data, wherein the respective first data represents a respective portion of a distributed computational graph” (Grounds 1 & 3, Claims 1-3, 5, 7).

Independent claim 1 recites “a first plurality of computer systems, wherein *each* respective computer system” of the plurality “comprises a memory that stores a respective first data, wherein the respective first data represents a respective portion of a distributed computational graph.”¹ Ex. 1001 [’424], cl. 1.

¹ Petitioner proposes that a “distributed computational graph” be construed as “[a] directed graph with each transformation comprising a node and the output

Petitioner fails to show how Barsness teaches “respective first data” in “each respective computer system” that “represents a respective portion of a distributed computational graph.” As will be explained below, while Petitioner identifies Barsness’s “operator graph” as the claimed “distributed computational graph,” *see, e.g.,* Pet., 20, even if this identification is entirely taken for granted for purposes of Petitioner’s argument, Barsness expressly locates its operator graph at its management system, and Petitioner never shows that other locations in Barsness’s network contain data representing portions of this graph.

Petitioner nevertheless contends that the claimed “first plurality of computer systems” is satisfied by: (1) Barsness’s “management system,” which Barsness discloses exactly one of (element 105), and (2) one or more “compute nodes” of Barsness’s Figure 1A (elements 130₁-130₄):

messages between transformations comprising edges.” Pet., 7–8. Patent Owner opposes this construction, and proposed that this term be given its plain and ordinary meaning in district court. Ex. 1019 [Patent-Owners-Preliminary-Disclosure] 3. Nevertheless, Patent Owner does not believe that any issues discussed in this Patent Owner’s Preliminary Response turn on the construction of this term.

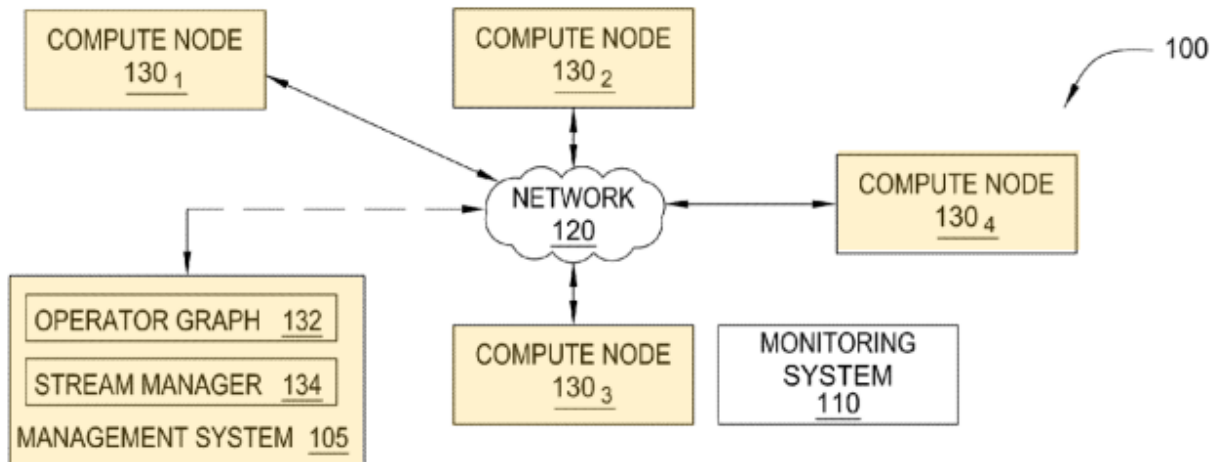


FIG. 1A

Pet., 14 (Petitioner’s highlighting).

Under Petitioner’s mappings, claim 1 requires that each respective computer system in Petitioner’s alleged plurality store “respective first data” that “represents a respective portion” of Barsness’s operator graph. Petitioner’s initial mapping does not show that requirement for either type of system in its alleged plurality. Petitioner points generally to “programming instructions,” “application data,” and storage containing “buffered stream data 260 and operator graph 132,” as alleged examples of the claimed “respective first data,” but Petitioner never shows that this data (allegedly stored at each respective system) represents a respective portion of the operator graph. That problem is especially apparent for the compute nodes: Barsness does not, for example, disclose that compute nodes store the Barsness operator graph or any portion of the Barsness operator graph, and Petitioner does not show how the compute nodes’ alleged “programming instructions,” “application data,” or

“buffered stream data 260” represent any portion of the operator graph. *See* Section II.A.1, *infra*.

For Barsness’s compute nodes, Petitioner therefore turns to a different compute-node theory, relying on “processing elements (PEs) 235.” *See, e.g.,* Pet., 23. This theory also fails because Petitioner never reconciles the PEs with the “first data” it previously identified in Barsness’s compute nodes, and also because Barsness’s PEs are not shown to store data representing Barsness’s operator graph. *See* Section II.A.2, *infra*. These defects identified in connection with Barsness’s claim 1 are not further addressed in Petitioner’s analysis of claims that depend from claim 1, or by the Siripurapu reference used in Petitioner’s Barsness-Siripurapu Ground 3. *See* Section II.A.3, *infra*.

1. Petitioner Does Not Consistently Identify The Claimed “respective first data.”

Although Petitioner’s alleged plurality involves systems of two distinct types—Barsness’s management system, on the one hand, and Barsness’s compute nodes, on the other, *see, e.g.,* Pet., 14—Petitioner is not initially clear about how it identifies the claimed “respective first data” within each member of its plurality.

Petitioner generally states, for example, that each Barsness compute node “comprises ‘memory 225’ (a *memory*) that ‘store[s]’ data, e.g., programming instructions and ‘application data’ (*stores a respective first data*),” Pet., 17 (italics in original), apparently identifying “e.g., programming instructions and ‘application

data” as the claimed “respective first data” at Barsness compute nodes. Petitioner then says that Barsness’s management system “comprises ‘memory 325’ (a memory) that stores data, e.g., ‘programming instructions and application data,’” Pet., 17, apparently also identifying general “programming instructions and application data” with the claimed “respective first data” at Barsness’s management system.

But Petitioner never shows that these generic “programming instructions” or “application data” represent any respective portion of Barsness’s operator graph, as Petitioner’s claim mapping would require. Having identified Barsness’s operator graph as the claimed “distributed computational graph,” Pet., 20, Petitioner needed to show that the “first data” stored at each respective computer system represents a respective portion of that graph. Petitioner’s general references to “programming instructions” and “application data,” Pet., 17–19, do not do so. Indeed, the Petition effectively admits this: after identifying generic “programming instructions” and “application data” with the claimed “first data” in connection with what Petitioner terms element [1.2.a] of claim 1 (“wherein each respective computer system of the first plurality of computer systems comprises a memory that stores a respective first data”), Pet., 17, Petitioner never returns to this generic alleged “first data” when addressing what Petitioner terms element [1.2.b] of claim 1 (“wherein the respective first data represents a respective portion of a distributed computational graph”). *See* Pet., 20–24.

Petitioner says in passing that “[*each computer system* also includes ‘storage’ 230/330 that stores data, e.g., buffered stream data 260 and operator graph 132,” Pet., 17—possibly suggesting that *each* member of Petitioner’s plurality stores both “buffered stream data 260” and Barsness’s “operator graph 132.” But Barsness does not disclose that either. As Dr. Malek explains:

Barsness discloses that its compute nodes may contain Barsness’s “buffered stream data 260,” and that its management system may contain Barsness’s “operator graph 132.” This is seen, for example, in Barsness’s Figure 2 (which Petitioner acknowledges illustrates a Barsness compute node, Pet., 15), which depicts “buffered stream data 260” within a box labeled “storage 230” on a Barsness compute node, and Barsness’s Figure 3 (which Petitioner acknowledges illustrates Barsness’s management system, Pet., 17), which depicts “operator graph 132” within a box labeled “storage 330” inside Barsness’s management system. But Barsness does not disclose that its management system stores “buffered stream data 260,” and Barsness does not disclose that its compute nodes store Barsness’s “operator graph 132.”

Ex. 2001 [Malek-Decl.] ¶ 38.

Barsness thus places two specific items in different systems: operator graph 132 in the management system, and buffered stream data 260 in the compute nodes. Petitioner relies on operator graph 132 for alleged “respective first data” in Barsness’s management system, arguing that the management system satisfies the

limitation because it stores the entire operator graph. *See* Pet., 24. But this theory does not apply to the compute nodes. Petitioner does not argue, and Barsness does not disclose, that the compute nodes store operator graph 132 (or indeed, any portion of the operator graph 132). Pet., 17, 20–24; *see also, e.g.*, Ex. 1004 [Barsness], Figs. 2–3; 9:43–50; 10:12–14, 48–56; Ex. 2001 [Malek-Decl.] ¶ 39. Nor does Petitioner argue that buffered stream data 260 represents any portion of the operator graph. *See* Pet., 20–24.

2. Petitioner Fails To Show That Barsness’s Compute Nodes Store “respective first data [that] represents a respective portion of a distributed computational graph.”

For Barsness’s compute nodes, Petitioner offers a new candidate for the claimed “respective first data”—PEs 235 allegedly stored in compute-node memory—and a new theory for how this data “represents a respective portion of” Barsness’s operator graph. Petitioner states:

For example, “memory 225 includes one or more processing elements (PEs) 235.” [Barsness], 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.

Pet., 23. This theory fails for two independent reasons.

First, Petitioner never reconciles this new identification of the claimed “respective first data” with the generalized “programming instructions and

‘application data’” or “buffered stream data 260” that Petitioner previously identified as “respective first data” at Barsness compute nodes. Pet., 17. When addressing the requirement that “the respective first data” represent a graph portion, however, Petitioner relies on PEs 235. Pet., 23. Claim 1’s reference to “the respective first data,” by contrast, is a reference back to its earlier recited “a respective first data”—*i.e.*, the **same thing** as the earlier-recited “first data.” *See, e.g., Convolve, Inc. v. Compaq Computer Corp.*, 812 F.3d 1313, 1321 (Fed. Cir. 2016) (“This reference to ‘the processor,’ referring back to the ‘a processor’ recited in preamble, supports a conclusion that the recited user interface is ‘operatively working with’ the **same** processor to perform all of the recited steps”) (emphasis added).

The Petition therefore had to explain how PEs 235 relate to the “programming instructions,” “application data,” or buffered stream data on which it previously relied. It does not. As Dr. Malek explains,

Barsness’s Figure 2 expressly indicates that its **processing elements** 235 are distinct from the storage 230 that contains the **buffered stream data** 260. Barsness’s statement that the “application data” cited by Petitioner “resid[es] in the **memory** 225” of Figure 2, Ex. 1004 [Barsness], 8:53-55, on the other hand, only indicates to the POSITA that this application data, whatever it is, resides in the same **memory** of the compute node that the **processing elements** 235 do:

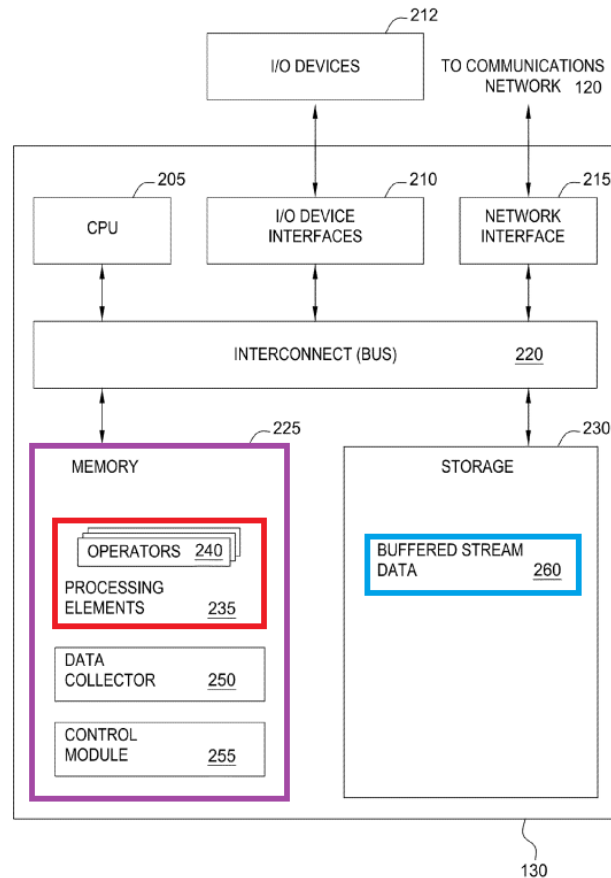


FIG. 2

This disclosure does not show that Petitioner’s cited “application data” is the *same thing* as the **processing elements** 235, or even that Petitioner’s cited application data is *stored in the same part of memory* as the **processing elements** 235. And Petitioner does not elsewhere or otherwise discuss how its earlier-identified “first data” relates to the processing elements. Accordingly, Petitioner has not shown, with the “first data” it identifies in compute nodes in connection with what it labels limitation [1.2.b], and the “first data” it previously identified in compute nodes in connection with what it labels limitation [1.2.a], how limitation [1.2.b] is allegedly taught by Barsness.

Ex. 2001 [Malek-Decl.] ¶¶ 43-44.²

Second, even if PEs 235 were treated as the claimed “first data,” Petitioner does not show that they “represent[] a respective portion of a distributed computational graph.” Petitioner has the representational relationship backwards: Barsness’s operator graph represents the PEs; the PEs do not represent the operator graph. As Dr. Malek explains,

Petitioner states, for example, that “[a]s 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.” Pet., 23. I do not agree. As a threshold matter, it is imprecise to say that each PE “is” a node in Barsness’s operator graph, as the verb “is” may misleadingly suggest literal identity between PEs and operator graph nodes, or literal inclusion of PEs within operator graph nodes. Barsness’s operator graph is an abstract data structure whose nodes *represent* PEs, and a POSITA might say that each PE in the memory of each Barsness compute node is *represented by* a node in Barsness’s operator graph.

To say that the PEs that literally exist in the memories of Barsness’s compute nodes “represent[] ... a portion of the operator graph” by virtue of the fact that they are represented by nodes within that graph is to reverse the relationship between the map (Barsness’s

² Here and elsewhere, labels of individual claim limitations refer to Petitioner’s labeling. Pet., viii-ix.

operator graph) and the territory (Barsness's PEs). Just as an English speaker would not refer to the Lincoln Memorial as data representing a portion of a map of Washington, D.C., a POSITA would not recognize Barsness's PEs as "data represent[ing] a portion" of the operator graph that refers to them. Because Petitioner does not identify any other basis on which Barsness's PEs could "represent[] a respective portion of" the operator graph, Petitioner has not shown how this limitation is met.

Ex. 2001 [Malek-Decl.] ¶¶ 46-47. Petitioner accordingly fails to show how this limitation is taught by Barsness.

3. Petitioner's Arguments Regarding Dependent Claims And The Barsness-Siripurapu Combination Do Not Cure These Defects.

Petitioner's arguments as to claims that depend from claim 1 do not address the deficiency identified above in connection with claim 1. *See generally* Pet., 49–60, 77–83. Similarly, Petitioner's Barsness-Siripurapu Ground 3 relies entirely on Barsness and Petitioner's Ground 1 argument for this limitation (and, in particular, Petitioner does not provide any separate analysis for this limitation as it might be taught by Barsness-Siripurapu). *See* Pet., 77–83. So although the analysis above focused on Petitioner's Ground 1, it applies without change to Petitioner's Ground 3. Accordingly, Petitioner's Grounds 1 and 3 fail as to all challenged claims that depend from claim 1.

B. Petitioner Fails To Show That Barsness Teaches “first data that ... identifies at least a second transformation pipeline” (Grounds 1 & 3, Claims 9-11, 13).

Independent claim 9 recites “a first plurality of computer systems, wherein *each* respective computer system” of the plurality “comprises a respective memory that stores a respective first data that ... *identifies* at least *a second transformation pipeline*.” Ex. 1001 [’424] cl. 9. Petitioner contends that “transformation pipeline” should be construed as “[a] progression of functions which each perform a transformation on a data stream.” Pet., 9; Ex. 1020 [Petitioners-Preliminary-Claim-Constructions] 1.³

As with claim 1, *see supra*, Petitioner identifies the claimed “first plurality of computer systems” as consisting of Barsness’s management system 105 and one or more Barsness compute nodes (*e.g.*, 130₁, 130₂). Pet., 62.

³ Patent Owner proposed that this term be given its plain and ordinary meaning. Ex. 1019 [Patent-Owners-Preliminary-Disclosure], 3.

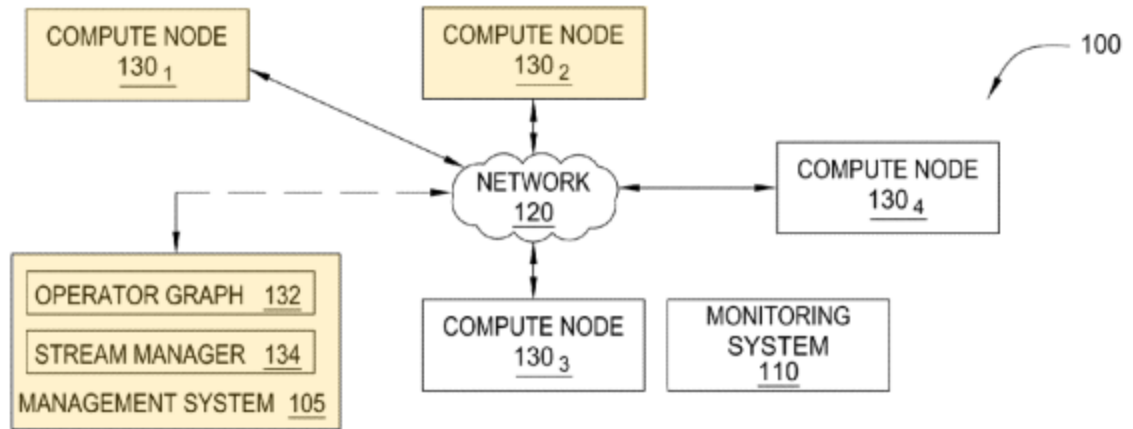


FIG. 1A

See Pet., 62 (Petitioner’s highlighting).

But Petitioner never provides a complete mapping of this limitation. For each member of its alleged plurality, Petitioner needed to identify the stored “first data,” the alleged “second transformation pipeline,” and how that data identifies that pipeline. Instead, Petitioner offers two incomplete theories. It first asserts that its claim 1 analysis also satisfies claim 9, although the cited analysis addresses a different limitation and does not provide any mapping to the elements of claim 9. Petitioner then offers a conditional theory under which a compute node is treated as the “second transformation pipeline.” But neither theory shows the limitation.

1. Petitioner’s Argument By Reference Fails.

Petitioner first implicitly attempts to identify a “second transformation pipeline” by incorporating its arguments about claim 1 into its arguments about this limitation of claim 9 by reference. *See, e.g.*, Pet., 62–63 (stating that “each of the management system 105 and compute nodes 130 ... stores respective programming

instructions, application data, and other data” and “[t]hat data meets the additional recited claim language,” citing Petitioner’s analysis of claim 1).

This cross-reference does not supply the missing claim 9 mapping. The cited claim 1 discussion addresses whether stored data “represents a respective portion of a distributed computational graph”; it does not identify, for each member of Petitioner’s claim 9 plurality, what stored data allegedly identifies which “second transformation pipeline.” Indeed, Petitioner cites its analysis of limitation [1.2.b], Pet., 20–24, while its claim 1 discussion of a “second transformation pipeline” appears elsewhere. *See* Pet., 24–27. Thus, the cross-reference identifies neither the alleged data, the alleged pipeline, nor the relationship between them.

It is not the Board’s place to piece together an argument that a petitioner only vaguely alludes to with references to other sections and statements like “[t]hat data meets the additional recited claim language.” Pet., 63; *cf.* 35 U.S.C. § 312(a)(3) (petition must identify, “with particularity,” how each challenged claim is alleged to be obvious); *Netflix, Inc. v. DivX, LLC*, 84 F.4th 1371, 1377 (Fed. Cir. 2023) (“[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. Ultimately, it is the petitioner’s burden to present a clear argument”). Petitioner’s argument by incorporation thus fails to show how Barsness teaches a “second transformation pipeline” in the sense of claim 9.

2. Petitioner's Argument That Barsness's Compute Node Is A "second transformation pipeline" Does Not Show How Petitioner's Own Construction Is Satisfied.

The Petition's only direct attempt to identify a "second transformation pipeline" in its claim 9 analysis is conclusory:

Similarly, if compute node 130₂ is viewed as the second transformation pipeline (*supra* Sections VII.B.3.d-VII.B.4), then at least that compute node (130₂) and the compute node from which it receives messages (130₁) each has memory storing data that identifies at least a second transformation pipeline (the pipeline of compute node 130₂). If compute node 130₃ is viewed as the second transformation pipeline, then it and 130₁ satisfy this element.

Pet., 65.

This conditional phrasing does not provide a complete mapping. It does not explain why compute node 130₂ or 130₃ is itself a "transformation pipeline," identify the particular stored data at each system that allegedly identifies that pipeline, or explain how that data performs the claimed identification. Its internal cross-reference to several pages of claim 1 analysis does not answer those questions. (Those sections address the transmission and processing of pipeline output messages; they do not identify, for each member of the claim 9 plurality, the stored data that identifies the alleged second pipeline. *See* Pet., 37–41, 65.) Here again, Petitioner is simply asking the Board to find that *some* review of Barsness and its claim 1 argument would lead to a finding that Barsness also teaches similarly worded

limitations as arranged in a completely different claim. Petitioner does not even directly state that Barsness's compute node 130₂ *is* the claimed "second transformation pipeline" in its application of Barsness to claim 9, let alone show it. Petitioner merely invites speculation about what might follow "*if* compute node 130₂ is viewed as the second transformation pipeline" Pet., 65. This is insufficient to meet an IPR petitioner's burden. *Netflix*, 84 F.4th at 1377; *see also, e.g., Microsoft Corp. v. FG SRC, LLC*, 860 F. App'x 708, 713 (Fed. Cir. 2021) (nonprecedential) ("It is not the Board's job 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.").

If the Board were to accept Petitioner's statement that "*if* compute node 130₂ is viewed as the second transformation pipeline" as an assertion by Petitioner that "compute node 130₂" in Barsness's Figures 1A and 1B literally *is* (or teaches) the "second transformation pipeline" to which Petitioner's claim 9 analysis applies, this theory remains undeveloped—for example, because Petitioner fails to explain why the arrangement of PEs on compute node 130₂ is a "second transformation pipeline" under Petitioner's proposed construction of that term.

Petitioner's proposed construction of "transformation pipeline" is "[a] progression of functions which each perform a transformation on a data stream." *See, e.g.,* Pet., 9. Barsness's Figure 1B depicts the compute node 130₂ as associated

with three processing elements: PE4, PE5, and PE6.

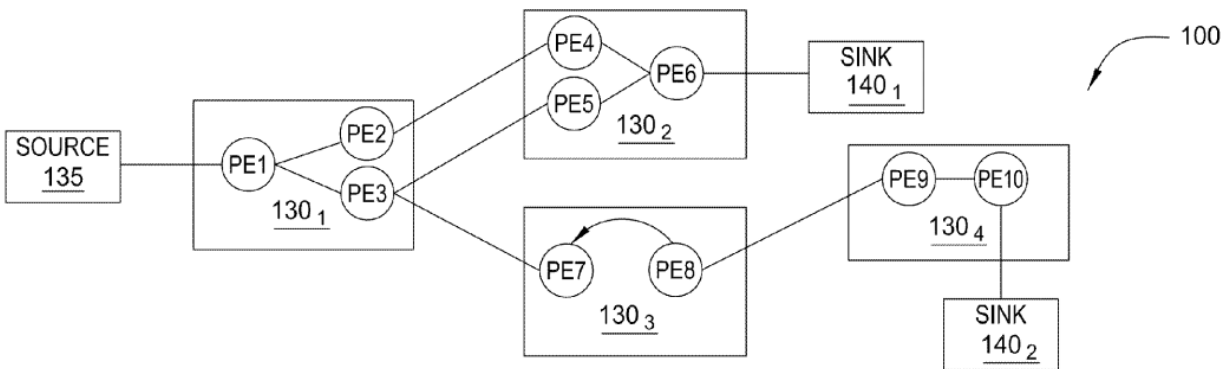


FIG. 1B

Ex. 1004 [Barsness], Fig. 1B. Barsness discloses that PE4 receives data “tuples” from PE2 within Barsness’s “compute node 130₁” and that PE5 receives data “tuples” from PE3 within Barsness’s “compute node 130₁,” and that data “tuples” are emitted from both PE4 and PE5 to PE6 within Barsness’s “compute node 130₂.”

Ex. 1004 [Barsness], 7:58-61. As an example of what PE6 might do with these tuples, Barsness suggests that “PE6 could be configured to perform a join operation for combining tuples received from PE4 and PE5.” *Id.*, 7:61-64.

But Petitioner offers no explanation for how a processing element that “combines” tuples received from two different processing elements can be part of a “progression of functions which each perform a transformation on *a* data stream,” as its construction requires. Ex. 2001 [Malek-Decl.] ¶ 52. It was Petitioner’s burden to explain its theory that PE6, which receives tuples from PE4 and PE5, performs a transformation on “*a* data stream” as its construction of “transformation pipeline”

requires. Petitioner failed to meet this burden.⁴

Petitioner does purport to offer a general analysis of individual compute nodes against its proposed construction of “transformation pipeline.” *See* Pet., 25–26. But its general—and conclusory—analysis fails to actually engage with the specifics of any of the compute nodes. When it comes to compute node 130₁, Petitioner’s alleged “first transformation pipeline,” for example, Petitioner states: “the progression of transformations performed by PE1, PE2, and PE3 in the compute node 130₁ is a first transformation pipeline.” Pet., 25. Petitioner’s passing acknowledgment of PE4, PE5, and PE6 is no more specific. *Id.*, 26. This *ipse dixit* analysis fails to offer a reasoned analysis that the tribunal can evaluate and that Patent Owner can fairly respond to.

3. Petitioner’s Arguments Regarding Dependent Claims And The Barsness-Siripurapu Combination Do Not Cure These Defects.

Petitioner’s arguments as to claims depending from independent claim 9 do

⁴ Petitioner correctly observes that a transformation pipeline “can include branching” configurations. *See* Pet., 25 n.2. Nevertheless, Petitioner fails to explain how its proposed construction, which requires the transformations of a transformation pipeline to each perform a transformation on “a” data stream, accommodates such configurations, let alone whether (or how) Barsness’s PEs are mapped to such a configuration.

not address the deficiencies identified here in connection with claim 9. Similarly, Petitioner’s analysis of this limitation in claim 9 in its Barsness-Siripurapu Ground 3 relies on Barsness alone. *See* Pet., 77–83. Accordingly, Petitioner’s Grounds 1 and 3 fail as to all challenged claims that depend from claim 9.

C. Petitioner Fails To Demonstrate That Barsness Teaches The “third computer system of the first plurality” Limitations (All Grounds, All Challenged Claims).

Claim 1 requires “a third computer system of the first plurality of computer systems ... 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.” Ex. 1001 [’424] cl. 1. Independent claim 9 recites a similar limitation—“a third computer system of the first plurality of computer systems ... 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,” *see* Pet., 67—but as to the issues identified below, Petitioner does not offer separate argument about this limitation as it appears in claim 9. *See id.* (Ground 1), 77 (Ground 2), 82–83 (Ground 3), 83–84 (Ground 4).

1. Petitioner’s Ground 1 Argument Fails.

The claim limitation at issue requires “a third computer system of the first

plurality of computer systems ... 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.” Petitioner’s Ground 1 mapping is as follows:

Claim element	Petitioner’s Mapping to Barsness
“third computer system of the first plurality”	Barsness’s management system <i>See, e.g., Pet., 44.</i>
“second plurality of computer systems”	“additional, unconfigured compute nodes / computer systems that are available to be configured and assigned by the management system when needed” in Barsness. <i>See, e.g., Pet., 42.</i>
“fourth computer system”	“an additional compute node” in Barsness. <i>See, e.g., Pet., 44.</i>
“first transformation pipeline”	“transformations” allegedly performed by PE1, PE2, and PE3 of Barsness’s compute node 130 ₁ . <i>See, e.g., Pet., 26.</i>
“second transformation pipeline”	“transformations” allegedly performed by PE4, PE5, and PE6 of Barsness’s compute node 130 ₂ . <i>See, e.g., Pet., 26.</i>

Thus, under its mapping, Petitioner must show that Barsness’s management system “is configured to execute software instructions” that cause one or more additional compute nodes “to execute software instructions that apply” either the “transformations” allegedly performed by Barsness’s compute node 130₁, or the “transformations” allegedly performed by Barsness’s compute node 130₂.

Petitioner does not do this. Petitioner begins by citing various disclosures

about the Barsness system, many of which implicate components of the system other than Barsness’s management system, while attributing all of the cited functionality to the Barsness management system—presumably because this is what Petitioner has mapped to the claimed “third computer system.” *See* Section II.C.1.a), *infra*. Even if all of the cited functionality is attributed to Barsness’s management system, however, the limitation is still not shown, because Petitioner’s argument at most shows that the management system is involved in *allocating resources* in Barsness. Petitioner does not show that the management system actually “caus[es] 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” as required. *See* Section II.C.1.b), *infra*.

a) *The Activity Petitioner Identifies In Barsness’s Management System Is Initiated By Barsness’s Monitoring System.*

Petitioner maps the claimed “third computer system” to Barsness’s management system, *see, e.g.*, Pet., 44, but in discussing Barsness’s management system, Petitioner repeatedly cites actions triggered by a separate component—Barsness’s monitoring system—that is not a part of Barsness’s management system. Indeed, Barsness’s monitoring system is not even included in what Petitioner identifies as the “first plurality of computer systems” of which a claimed “third computer system” must be a part:

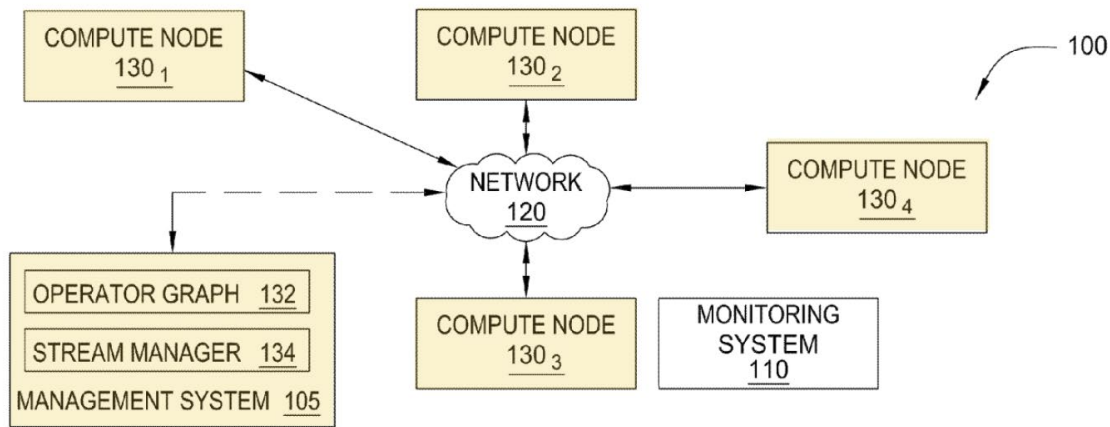


FIG. 1A

Pet., 14 (Ex. 1004 [Barsness], Fig. 1A, Petitioner’s highlighting identifying alleged members of the “first plurality of computer systems,” and expressly excluding Barsness’s monitoring system).

Petitioner begins, for example, by arguing that “Barsness specifically teaches that its management system responds to performance issues by provisioning additional compute nodes and assigning or reassigning processing elements among those nodes.” Pet., 45. According to Petitioner, “if the management system detects performance issues associated with compute node 130₁, Barsness teaches provisioning an additional compute node (e.g., compute node 130_x) and assigning to it the same processing elements used to implement the first transformation pipeline, such that the new compute node begins executing that pipeline as well.” Pet., 44. Petitioner’s proposal is illustrated by a modified version of Figure 1B with a new compute node “130_x” that appears to play the role of the claimed “fourth computer

system”:

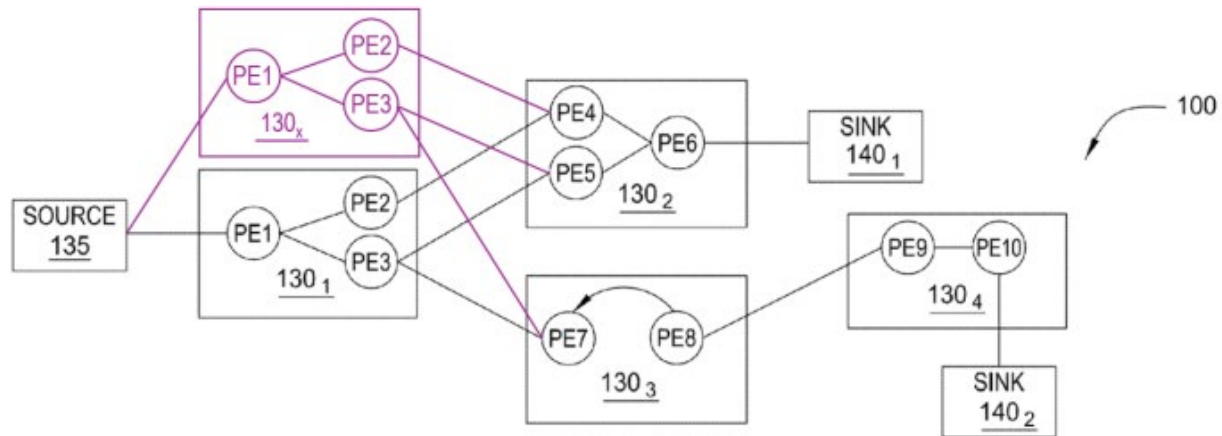


FIG. 1B

Pet., 44–45 (Fig. 1B, modified by Petitioner to include a compute node “130_x”).

As a threshold matter, however, Barsness does not disclose that its “management system” is tasked in the first instance with “detect[ing] performance issues associated with compute nodes.” Pet., 44. Barsness discloses that its “**monitoring system** 110 determines a need based on a detection of performance issues with a computer node 130 or the processing elements 235 in a computer node 130.” Ex. 1004 [Barsness], 10:28-34; *see also id.*, 10:34-39 (teaching that “monitoring system 110” may issue instructions to “improve the data processing performance of the stream application”). Elsewhere, Petitioner’s analysis of claim 7 correctly identifies the direct role of Barsness’s monitoring system in detecting “performance issues” in Barsness. Pet., 59 (referring to what should happen “if the **monitoring system** detects ‘performance issues’ with a compute node”).

As Petitioner indirectly notes, Barsness also discloses that in some embodiments, *after* its monitoring system has detected performance issues or other reasons for potentially reconfiguring the Barsness network, Barsness’s monitoring system may respond by instructing the Barsness *management system*—*i.e.*, Petitioner’s alleged “third computer system”—to “spawn more process nodes” or “move processing elements ... from one computer node ... to another.” Ex. 1004 [Barsness], 10:28-39; *see also* Pet., 46 (citing these disclosures).

Petitioner thus demonstrates, at most, that Barsness’s management system *plays a role* in “spawning more process nodes” or “mov[ing] processing elements” in Barsness. *See* Pet., 46-47. It does not necessarily follow, however, that Barsness’s management system is “configured to execute software instructions that *cause*” a newly spawned or newly moved node to begin applying Petitioner’s mapped pipelines.

- b) *Petitioner Does Not Show That Barsness’s Management System Executes Software Instructions That Cause A Fourth Computer System To Execute Software Instructions Applying The First Or Second Transformation Pipeline.*

The claim requires a specific causal chain: the claimed third computer system must execute software instructions that cause “a fourth computer system” (claim 1) or “the second plurality of computer systems” (in claim 9) to execute software instructions applying the first or second transformation pipeline. At most, Petitioner shows that Barsness’s management system can, at the instruction of other

components of the Barsness system, provision a new compute node with PEs whose structure mimics that of a previously existing compute node. Petitioner suggests that should a new compute node then actually begin executing software instructions that apply the mapped pipeline, it must be doing so because of software instructions executed by the management system—but Petitioner does not show this.

Petitioner’s theory is that Barsness teaches that “its management system will 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 130₁.” Pet., 46-47. Petitioner illustrates that theory by modifying Barsness’s Figure 1B to insert a new compute node 130_x, which Petitioner treats as the claimed “fourth computer system.”

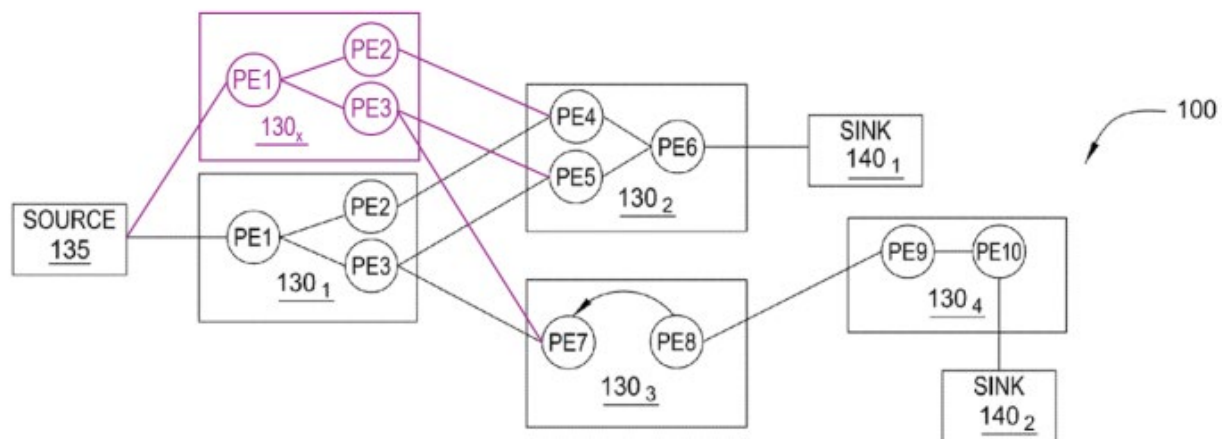


FIG. 1B

Pet., 44–45 (Fig. 1B, modified by Petitioner to include a compute node “130_x”),

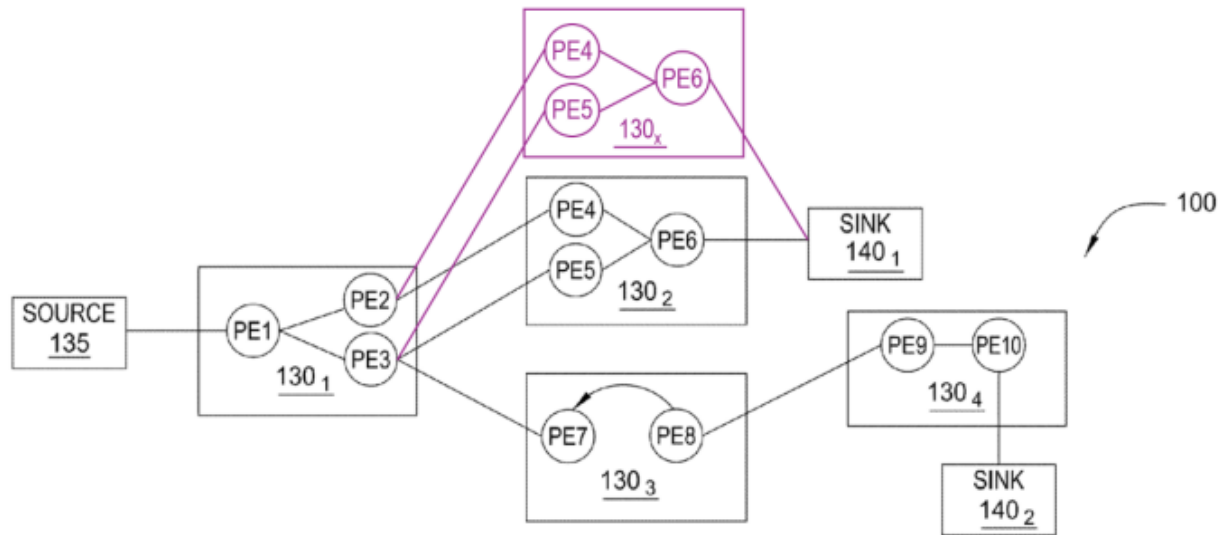


FIG. 1B

Pet., 48 (modified by Petitioner).

In the modified figures, Petitioner places on the new node the PEs that it maps to the first or second transformation pipeline. Pet., 26, 44–48. Thus, to satisfy this limitation, Petitioner must show that Barsness’s management system executes software instructions that cause the new compute node 130_x to execute software instructions applying the PE1/PE2/PE3 pipeline or the PE4/PE5/PE6 pipeline.

The modified figures illustrate Petitioner’s theory, but do not establish the required causal step. With Petitioner’s mappings, the claim does not merely require the management system to provision or configure a node that can apply a transformation pipeline; it would require the management system to execute software instructions that cause the new node to execute software instructions applying that pipeline. Petitioner assumes that provisioning or duplicating a node

supplies that step, but does not explain how the cited Barsness disclosures show that the *management system* causes the new node to execute the mapped PE1/PE2/PE3 or PE4/PE5/PE6 pipeline. As Dr. Malek explains:

A POSITA would not read Barsness's provisioning or duplication disclosures cited by Petitioner as satisfying the claimed causal relationship. *See* Pet., 45-49 (relying on Ex. 1004 [Barsness], 8:43-52, 10:26-39, 14:30-57). Those disclosures concern spawning, provisioning, duplicating, moving, or configuring compute-node resources. In embodiments where the management system does this, it does so at the instruction of another component of the Barsness system. These disclosures do not identify software instructions executed *by the management system* to cause any newly provisioned node to execute any functions performed in Petitioner's identified PE1/PE2/PE3 or PE4/PE5/PE6 "pipelines."

Ex. 2001 [Malek-Decl.] ¶ 61.

Petitioner therefore does not bridge the gap between provisioning or configuring a node and causing that node to execute the mapped pipeline. Because Petitioner does not show that Barsness's provisioning or duplication establishes the claimed causal relationship, it has not shown that this limitation is satisfied.

2. Petitioner's Grounds 2-4 Arguments Also Fail.

Petitioner's Barsness-Chakradhar Ground 2 relies upon Chakradhar solely for disclosures relating to limitations that are not at issue in the analysis above, and in particular, does not argue that Chakradhar teaches anything relevant to the limitation

at issue (what Petitioner calls limitation [1.7]). *See generally* Pet., 73–77.

Accordingly, Petitioner’s Ground 2 does not cure the defects identified in Section II.C.1, *supra*.

Petitioner’s Barsness-Siripurapu Ground 3 does not cure this defect either. Petitioner relies on Siripurapu only for what Petitioner refers to as supplemental detail about “scaling and multiple data streams.” Pet., 77. Petitioner’s discussion is correspondingly general: it argues that a POSITA would have been motivated “to configure the Barsness infrastructure to scale the stream-processing application by adding additional, duplicate nodes to handle additional data streams from other applications,” Pet., 81, and “to provision additional compute node(s) to handle increased data volume / processing needs,” *id.*, 82. But those scaling arguments do not address the limitation missing from Ground 1. They do not identify any Siripurapu component mapped to the claimed “third computer system,” and they do not show software instructions that cause a fourth computer system to execute software instructions applying Petitioner’s mapped first or second transformation pipeline. Accordingly, Petitioner’s Ground 3 does not cure the defects identified in Section II.C.1, *supra*.

Ground 4 adds nothing material. Petitioner does not separately analyze the relevant limitation for its three-reference combination; it merely asserts that Chakradhar and Siripurapu are “fully compatible with and complementary to one

another” and that a POSITA would have implemented both in Barsness. Pet., 83–84. That conclusory combination rationale does not identify software instructions executed by the mapped third computer system that cause a fourth computer system to execute software instructions applying the first or second transformation pipeline. Ground 4 therefore does not cure the defects identified in Section II.C.1, *supra*.

D. Petitioner Does Not Show A Motivation To Combine Barsness With Chakradhar With A Reasonable Expectation Of Success (Grounds 2, 4, All Challenged Claims).

Petitioner’s Grounds 2 and 4 propose combinations involving Barsness and Chakradhar. Petitioner’s Ground 4 rationale for combining Chakradhar with Barsness incorporates its Ground 2 rationale with no substantive modification. *See generally* Pet., 83–84. For this reason, although this section focuses on Petitioner’s Ground 2 rationale, this section applies to both Grounds 2 and 4.

Petitioner’s rationale for combining Barsness with Chakradhar is premised upon benefits supposedly apparent to the POSITA from the use of “Chakradhar’s ability to update routing behavior ‘on-the-fly’ and ‘atomically’ across operators,” which Petitioner argues “supports fast reconfiguration, scalability, and fault tolerance.” Pet., 71–72. This updating ability is provided in Chakradhar by a component called a “controller” that Chakradhar uses to create and update data structures called “route-maps” that are then installed at the relevant operators in Chakradhar’s network.

But Petitioner’s proposed combinations do not involve Chakradhar’s controller: Petitioner posits only that if the data structures *created by* Chakradhar’s controller—*i.e.*, “route-maps”—were somehow stored at “compute nodes” in Barsness, then the resulting system would teach various limitations of claim 1.⁵ *See generally* Pet., 72 (describing the combination as “incorporating Chakradhar’s route-maps into Barsness” and “[a]pplying Chakradhar’s route-maps to Barsness”), 75 (identifying “route-maps” with the “first data” of claim 1 and arguing that they “represent[] a respective portion of a distributed computational graph”).

There is thus a fundamental mismatch between Petitioner’s benefits-based rationale for a motivation to combine, which crucially relies on functionality provided by Chakradhar’s *controller*, and Petitioner’s actual combination, which does not identify Chakradhar’s controller (or any substitute component) as part of the combination, or otherwise explain how route-map creation and updates occur in the combination. *See, e.g.*, Pet., 71–76. This gap—which the Petition does not acknowledge, let alone attempt to fill—prevents Petitioner from showing a motivation to combine. *See, e.g., Medtronic, Inc. v. Teleflex Innovations S.A.R.L.*,

⁵ Petitioner’s Ground 2 does not provide any analysis specific to independent claim 9, Pet., 77, and Petitioner’s Ground 4 does not specifically analyze any claim. Pet., 83.

69 F.4th 1341, 1351 (Fed. Cir. 2023) (affirming finding of no motivation to combine where the alleged benefit of a secondary reference was not attributable to the feature petitioner proposed to incorporate, finding that to “countenance motivation arguments based on ... features of references that happen to exhibit a benefit for altogether different reasons” would be “a recipe ... ripe for hindsight abuse”); *Intelligent Bio-Systems, Inc. v. Illumina Cambridge, Ltd.*, 821 F.3d 1359, 1368 (Fed. Cir. 2016) (affirming finding of no motivation to combine when petitioner’s alleged motivation posited benefits that had not been shown to exist in the combination).

As Dr. Malek explains:

Even if Petitioner were to propose that its combination of Barsness and Chakradhar could somehow incorporate a literal copy of Chakradhar’s controller into the combined system, for purposes of generating Chakradhar’s “route-maps” inside the combined system—and, to be clear, Petitioner does not actually make such a proposal—Petitioner does not suggest answers to such basic questions as where this component would be located (*e.g.*, whether in a modified version of Barsness’s management system, or in a modified version of Barsness’s monitoring system, or in some other place), let alone address why a POSITA would be motivated to make the design choices that Petitioner leaves unspecified.

Ex. 2001 [Malek-Decl.] ¶ 66.

The same gap defeats a reasonable expectation of success. Petitioner relies on Chakradhar’s route-maps, but its combination does not include Chakradhar’s

controller—the component that creates and updates the route-maps and orchestrates their installation. *See, e.g.*, Ex. 1005 [Chakradhar] ¶¶ 127–133. Nor does Petitioner explain what component in the combined system would perform that work. Petitioner therefore has not shown that a POSITA would have had a reasonable expectation of successfully implementing the proposed Barsness-Chakradhar system. Ex. 2001 [Malek-Dec1.] ¶ 67.

1. Chakradhar’s “Route-Maps” Are Created And Updated By Chakradhar’s Controller.

Petitioner’s rationale for combining Barsness with Chakradhar depends on “Chakradhar’s ability to update routing behavior ‘on-the-fly’ and ‘atomically’ across operators,” which Petitioner says “supports fast reconfiguration, scalability, and fault tolerance.” Pet., 71–72. In Chakradhar, however, that functionality is provided by the controller—not the route-maps alone.

This point is evident from the Chakradhar disclosure Petitioner cites:

By applying novel heuristics on the topology performance (feedback) metrics, advantageously *the controller determines efficient route-maps*, which encode tuple-routing information and also the topology link structure. 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.

Ex. 1005 [Chakradhar] [0054] (cited by Pet., 72).

The central role of Chakradhar’s controller in providing the functionality cited by Petitioner is repeatedly indicated elsewhere in Chakradhar. Chakradhar discloses that its “**controller** periodically collects metrics from the system” and “[b]ased on the observed bottlenecks, the **controller** computes new route-maps that minimize the maximum network and CPU utilization” and “also determines route maps that tune complex all-reduction topologies.” *Id.*, [0069]. Chakradhar confirms the controller’s role in its section entitled “On the Fly Topology Modification,” which describes the “light-weight, fault-tolerant protocol” cited by Petitioner:

So far we have described methods for dynamically changing key-space assignment and overlay topologies. Another novelty in methods according to the present disclosure lies in the method for **updating routing information** at operators to reflect the **changes proposed by the controller**.

Ex. 1005 [Chakradhar] [0121]; *see also* Pet., 70 (citing Ex. 1005 [Chakradhar] [0122]).

Subsequent paragraphs describe the steps of this protocol and confirm the controller’s central role. The protocol involves “two components: a **controller** (part of the scheduler) and a spout.” *Id.*, [0126]. The controller “creates new route[-]maps for each involved operator” (*id.*), stores them in local state, sends the new route-map message to the spout, receives route-installation confirmation, durably stores the new topology route-maps, sends an activation message, and marks the protocol

complete after receiving confirmation that the new routes have been activated. *Id.*, [0127]-[0133]; Ex. 2001 [Malek-Decl.] ¶ 69.

Other disclosures go into detail about how Chakradhar's controller computes its "route maps." The controller uses resource-assignment calculations based on inputs such as "CPU capacities of nodes," "network bandwidths (in and out) of all node-pairs," and "system throughput." *See id.*, [0087] & [0090]-[0093]. Chakradhar further explains that the controller can estimate network bandwidths from information like "the rate of tuple-acks emitted by different nodes in the past time-windows," and that the output of the resource-assignment problem is the route-map. *Id.*, [0099], [0101].

Dr. Malek confirms the key distinction Petitioner ignores: Chakradhar's controller creates and updates the route-maps, and the operators "maintain" those route-maps only in the sense that they store and consult them:

Chakradhar—including the portions relied upon by Petitioner—confirms that route-maps are created and updated by the controller. The controller determines efficient route-maps using topology-performance feedback metrics, computes new route-maps based on observed bottlenecks, and controls the protocol for updating routing information at operators. Ex. 1005 [Chakradhar], [0054], [0069], [0121]-[0133]. Chakradhar also describes the controller-side computations used to create route-maps, including resource-assignment calculations based

on inputs such as CPU capacities, network bandwidths, and system throughput. *Id.*, [0087], [0090]-[0093], [0101].

A POSITA would therefore not understand Chakradhar's statement that each operator "maintains a route-map" to mean that the operators create or update those route-maps. Ex. 1005 [Chakradhar], [0122]. In context, "maintains" means that each operator stores and consults a route-map supplied through Chakradhar's controller-based process. Thus, although Petitioner repeatedly relies on Chakradhar's operators "maintaining route-maps," Pet., 69, 70, 73, 75, that disclosure does not supply the missing controller functionality in Petitioner's Barsness-Chakradhar combination.

Ex. 2001 [Malek-Decl.] ¶¶ 71–72.

Thus, Petitioner has not explained how its combination obtains the asserted benefits from route-maps divorced from the controller-driven process that creates, updates, and provides them to operators. The Petition relies on Chakradhar's operators "maintaining" route-maps, but that disclosure shows only that in unmodified Chakradhar, Chakradhar's operators store and consult route-maps that are supplied through a controller-based process. Such disclosures do not explain how the functionality of Petitioner's alleged benefits-based rationale—"Chakradhar's ability to update routing behavior 'on-the-fly' and 'atomically' across operators," *see, e.g.*, Pet., 71—even appears in Petitioner's combined Barsness-Chakradhar system.

2. Petitioner Does Not Show A Reasonable Expectation Of Success Because It Does Not Explain How Route-Maps Are Created Or Updated In Its Combination.

Petitioner also fails to show a reasonable expectation of success. Petitioner's combination depends on route-maps existing at Barsness compute nodes and being available for use by Barsness operators. But Petitioner never explains how those route-maps would be created, updated, or injected in the proposed Barsness-Chakradhar combination. That omission is not a minor implementation detail. In Chakradhar, route-maps do not appear on operators by themselves; they are created, computed, updated, and injected through the controller-based process discussed above. *See* Section II.D.1, *supra*. That controller, however, is not a part of the combination and, consequently, Petitioner does not show why a POSITA would have had a reasonable expectation of success in executing the combination.

Petitioner's Ground 2 relies on Chakradhar's "route-maps." *See, e.g.,* Pet., 69 (arguing that "Chakradhar expressly ties the use of route-maps to concrete technical benefits"), 71 (arguing that a POSITA "would have been motivated to incorporate Chakradhar's route-map mechanism"), 75 (arguing that "route-maps are first data as recited in claim 1"). Petitioner says the combination would include Barsness's cloud-based stream-processing cluster and Chakradhar's teaching that operators store and maintain route-maps. Pet., 73. That does not explain the combination. In Chakradhar, the operators do not create the route-maps that they store. Chakradhar's

controller creates, computes, updates, and injects those route-maps through the controller-based protocol discussed above. *See* Section II.D.1, *supra*; Ex. 1005 [Chakradhar], [0126]-[0134].

Petitioner's element-by-element analysis repeats the same omission. Petitioner quotes Chakradhar's disclosures that "[e]very operator chooses a destination operator for its outgoing tuples based on route-maps," Pet., 75 (quoting Ex. 1005 [Chakradhar], [0069]), and that "[e]ach operator maintains a route-map that specifies the routes on which output messages should be sent after processing incoming messages," *id.* (quoting Ex. 1005 [Chakradhar], [0122]). Petitioner also states that route-maps are "injected" into the operators "for fast topology re-optimization." *Id.* (quoting Ex. 1005 [Chakradhar] [0010]). But Petitioner never accounts for the controller functionality that makes those disclosures work in Chakradhar. It never explains what components in the combined system would create the route-maps, update them, send them to the operators, or supply the inputs needed to compute them.

Dr. Malek confirms that Petitioner's reliance on operators "maintaining" route-maps does not fill that gap:

In Chakradhar, "maintains" means that each operator stores and consults a route-map supplied through Chakradhar's controller-based process. It does not mean that the operators create or update the route-maps. Ex. 1005 [Chakradhar], [0122]. Nor does Petitioner explain that

Barsness's compute nodes would create or update the route-maps they supposedly store. In particular, and as one example, Petitioner does not explain how Barsness compute nodes would have access to the inputs that Chakradhar's controller uses to compute route-maps, or how route-maps would be computed if some other component and other inputs are used to perform the work of Chakradhar's controller in Petitioner's combination.

Ex. 2001 [Malek-Decl.] ¶ 74.

Petitioner therefore has not shown that a POSITA would have had a reasonable expectation of successfully implementing the proposed Barsness-Chakradhar combination. An obviousness theory must identify a specific combination of prior art and provide a "clear[] explan[ation]" of "how the combination of the two references [is] supposed to work." *PersonalWeb Techs., LLC v. Apple, Inc.*, 848 F.3d 987, 994 (Fed. Cir. 2017). Petitioner fails to do this here: it relies on placing Chakradhar's route-maps inside Barsness, but never explains *how* the combined system would create, update, inject, or compute those route-maps; Petitioner's combination omits the controller-based process that performs that work in Chakradhar, but provides no replacement for it. Petitioner has therefore not shown how a POSITA would have expected the proposed combination to work, let alone to obtain the on-the-fly updating, fast reconfiguration, scalability, and fault-tolerance benefits on which Petitioner relies.

E. Petitioner Does Not Show A Motivation To Combine Barsness With Siripurapu Or A Reasonable Expectation of Success (Grounds 3 & 4, All Challenged Claims).

Petitioner’s Grounds 3 and 4 propose combinations involving Barsness and Siripurapu. Petitioner’s Ground 4 rationale for combining Barsness with Siripurapu incorporates its Ground 3 rationale with no substantive modification. *See generally* Pet., 83–84. For this reason, although this section focuses on Petitioner’s Ground 3 rationale, it applies to both Grounds 3 and 4.

Ground 3 is argued as a contingent fallback. Petitioner advances it only “[t]o the extent PO argues that Barsness provides insufficient detail on its scaling and multiple data streams.” Pet., 77. That hedge recurs in the combination analysis itself, where Petitioner argues motivation only “to the extent Barsness alone did not teach doing so.” Pet., 81. Although advancing arguments in the alternative is not in and of itself improper, Petitioner’s framing is telling—the ground fails on its own terms. First, if, as Petitioner contends, Barsness already performs the very scaling Siripurapu is offered to supply, then the combination Petitioner proposes solves no existing problem in Barsness and offers no improvement to Barsness’s system. Second, Petitioner never explains how the asserted combination would actually operate.

Petitioner contends that a POSITA would have been motivated to combine the references to add worker nodes that “accommodate increased computational

requirements, additional data streams, increased data stream flows, and the like.” Pet., 80 (quoting Ex. 1007 [Siripurapu] [0453]). But Petitioner does not explain why Barsness would benefit from that teaching; instead, Petitioner repeatedly asserts that Barsness already does the same thing. Petitioner asserts that Siripurapu’s teachings “reinforce the teachings in Barsness regarding ‘replicating processing elements on multiple nodes and load balancing among them’ . . . and spawning/provisioning more compute nodes as needed.” Pet., 80–81 (quoting Ex. 1004 [Barsness], 3:19–22). And Petitioner grounds its entire reasonable-expectation-of-success theory on the premise that “Barsness already discloses a scalable cloud-based computing infrastructure, where compute nodes can be dynamically provisioned/duplicated to handle data processing needs.” Pet., 82.

If Barsness already discloses the scaling functionality on which Ground 3 depends—as Petitioner says it does—then Siripurapu offers only redundancy. And if Barsness does not adequately teach that functionality, then Ground 3’s reasonable-expectation-of-success theory, which circularly rests entirely on Barsness’s own scalability (Pet., 82), collapses.

Either way, Petitioner has not shown a POSITA would be motivated to combine the two references. A POSITA would not have been motivated to add a feature that provided no benefit, and Petitioner has not identified any incremental benefit that would have motivated the proposed combination. *See Virtek Vision Int’l*

ULC v. Assembly Guidance Sys., 97 F.4th 882, 887 (Fed. Cir. 2024) (“[T]here must exist a motivation to combine various prior art references in order for a skilled artisan to make the claimed invention.”); *Hulu LLC v. Sound View Innovations LLC*, IPR2018-00582, Paper 34, 20–21 (Aug. 5, 2019) (informative) (rejecting combination where it was unclear why a modification would be a “good idea”); *In re Schweickert*, 676 F. App’x 988, 995 (Fed. Cir. 2017) (nonprecedential) (remanding when it was not explained how “Birrell would have ‘benefitted from the advantages of ... Cunniff’s semaphore mechanism’”).

Nor would a POSITA be motivated to add a redundant feature. *R.J. Reynolds Vapor Co. v. Fontem Holdings I B.V.*, IPR2016-01692, Paper 45, 27 (Mar. 2, 2018) (final written decision finding no motivation to combine; “[b]ecause [the first reference] already discloses a reliable means for transporting liquid from a liquid supply to a heating element, we do not find that one of ordinary skill in the art would have been motivated to use the technique of [the second reference] to improve [the first]”); *Apple Inc. v. Cellular Comm’ns Equipment, LLC*, IPR2015-00576, Paper 7, 12 (June 12, 2015) (denying institution where “Petitioner ... fails to provide a specific explanation why [the primary reference] did not already accomplish [the asserted benefits of the combination] by itself.”).

Even assuming a POSITA saw some abstract benefit in adding Siripurapu’s scalability teachings to Barsness, Petitioner never explains how the resulting

combination would operate. While Petitioner analogizes “the management system of Barsness” to “the conductor of Siripurapu,” Pet., 81, Petitioner does not say whether Siripurapu’s conductor would replace Barsness’s management system, operate beneath it, or operate alongside it. *See* Pet., 80–82. But an obviousness theory must identify a specific combination of prior art and provide a “clear[] explan[ation]” of “how the combination of the two references was supposed to work.” *PersonalWeb*, 848 F.3d at 994. Instead of such an explanation, Petitioner relies on generalized assertions that the references are “complementary,” “directly align[],” and would have been “naturally combined.” Pet., 80–82. But such conclusory reasoning is insufficient to establish obviousness. *See, e.g., ActiveVideo Networks, Inc. v. Verizon Commc’ns, Inc.*, 694 F.3d 1312, 1327 (Fed. Cir. 2012).

Accordingly, Petitioner fails to prove a reasonable likelihood of prevailing on Grounds 3 and 4.

III. CONCLUSION

For the foregoing reasons, institution should be denied.

Respectfully submitted,

Dated: June 22, 2026

By: /Kenneth J. Weatherwax/
Kenneth J. Weatherwax (Reg. No. 54,528)
Parham Hendifar (Reg. No. 71,470)
Nathan Lowenstein (*pro hac vice* to be filed)
Dennis Courtney (*pro hac vice* to be filed)
Colette Woo (*pro hac vice* to be filed)
LOWENSTEIN & WEATHERWAX LLP

CERTIFICATE OF COMPLIANCE WITH TYPE-VOLUME LIMITS

This Patent Owner's Preliminary Response ("POPR") consists of 8,814 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: June 22, 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 2001

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

Andrew M. Mason
Frank Morton-Park
Todd M. Siegel
Samuel B. Thacker

andrew.mason@klarquist.com
frank.morton-park@klarquist.com
todd.siegel@klarquist.com
samuel.thacker@klarquist.com
Msft-Qomplx@klarquist.com

Respectfully submitted,

Dated: June 22, 2026

By: /Michael Cosley/

Michael Cosley