

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

MICROSOFT CORPORATION,
Petitioner,

v.

PARTEC CLUSTER COMPETENCE CENTER GMBH,
Patent Owner.

IPR2025-00318
Patent No. 11,537,442

REPLY DECLARATION OF DARRELL LONG

MICROSOFT EXHIBIT 1030
Microsoft v. Partec
IPR2025-00318

TABLE OF CONTENTS

	Page
I. INTRODUCTION AND ENGAGEMENT	4
II. BACKGROUND AND QUALIFICATIONS	4
III. MATERIALS CONSIDERED AND INFORMATION RELIED UPON REGARDING '442 PATENT	4
IV. UNDERSTANDING OF PATENT LAW	7
V. LEVEL OF ORDINARY SKILL IN THE ART	8
VI. THE '442 PATENT	9
A. Ground 1	9
1. Claim 1 Is Obvious And Patent Owner's Arguments Fail	9
a) Lippert-Budenske Teaches Distributing And Redistributing Sub-Tasks To Both Computation Nodes And Booster Nodes	9
b) Dr. Kaeli's Reading Of Budenske Is Incorrect.....	12
2. Lippert-Budenske Enables The Claimed Invention Of The '442 Patent.....	15
a) Budenske's Teachings	15
b) Kwok's (EX1007) Successful Implementation Of Budenske.....	21
c) Additional Enabling Disclosures In the State Of The Art	27
3. A POSITA Would Have Combined Lippert And Budenske	31
4. Budenske Generates In Real-Time A Further Distribution Of Sub-Tasks Using Information Relating To The Processing Of The Plurality Of Sub-Tasks	33

5. Budenske’s Sub-Tasks Are Information Relating
To The Processing Of The Plurality Of Subtasks.....36

B. Claim 3 37

C. Claim 5 41

1. Lippert-Budenske Renders Obvious
Computation Node And Booster Node Daemons.....41

D. Claim 9 42

VII. GROUND 2 42

A. Claim 3 42

B. Claim 5 43

C. Claim 9 45

D. A POSITA Would Have Been Motivated To
Combine Kambatla With Lippert-Budenske.....45

I, Darrell Long, do hereby declare as follows:

I. INTRODUCTION AND ENGAGEMENT

1. I have been retained as an independent expert on behalf of Microsoft Corp. in connection with the above-captioned *Inter Partes* Review (“IPR”) proceeding to provide my analyses and opinions on specific technical issues related to U.S. Patent No. 11,537,442 (hereinafter “the ’442 patent”).

2. This Reply Declaration responds to certain points raised in the Declaration of Dr. David Kaeli (EX2021).

3. I filed a previous declaration (the “Initial Declaration,” EX1003), dated December 20, 2024, in this matter.

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

II. BACKGROUND AND QUALIFICATIONS

5. I incorporate by reference the description of my background and qualifications presented in my Initial Declaration and my curriculum vitae appended thereto as Appendix A. *See* EX1003, ¶¶4-11, Appendix A.

III. MATERIALS CONSIDERED AND INFORMATION RELIED UPON REGARDING ’442 PATENT

6. In preparing this declaration and arriving at my opinions, I have reviewed the following materials bearing Exhibit Nos. that I understand are being

referenced in the above-captioned IPR proceeding. In addition, I have reviewed the declaration of Dr. Kaeli, dated September 30, 2025 (EX2021), as well as all documents and exhibits cited therein.

LIST OF EXHIBITS

No.	Description
1001	U.S. Patent No. 11,537,442 (“the ’442 patent”)
1002	File History of U.S. Patent No. 11,537,442
1003	Declaration of Darrell Long, dated December 20, 2024
1004	U.S. Patent App. Pub. No. U.S. 2013/0282787 (“ Lippert ”)
1005	John R. Budenske et al., <i>A Method for the On-Line Use of Off-Line Derived Remappings of Iterative Automatic Target Recognition Tasks onto a Particular Class of Heterogeneous Parallel Platforms</i> , The Journal of Supercomputing, Vol. 12, No. 4 (Oct. 1998) (“ Budenske ”)
1006	U.S. Patent App. Pub. No. U.S. 2018/0074855 (“ Kambatla ”)
1007	Yu-Kwong Kwok et al., <i>A Semi-Static Approach to Mapping Dynamic Iterative Tasks onto Heterogeneous Computing Systems</i> , Journal of Parallel & Distributed Computing (2005) (“ Kwok ”)
1008	Mitchell D. Theys et al., <i>Mapping Tasks onto Distributed Heterogeneous Computing Systems Using a Genetic Algorithm Approach</i> , in <i>Solutions to Parallel and Distributed Computing Problems</i> (A. Zomaya, F. Ercal & S. Olariu eds., 2001) (“ Theys ”)
1009	U.S. Patent No. 3,496,551 (“ Driscoll ”)
1010	Yuan-Chieh Chow and Walter H. Kohler, <i>Models for Dynamic Load Balancing in a Heterogeneous Multiple Processor System</i> , IEEE Transactions on Computers, Vol. C-28, No. 5 (May 1979) (“ Chow ”)

No.	Description
1011	David M. Nicol and Paul F. Reynolds, Jr., <i>Optimal Dynamic Remapping of Data Parallel Computations</i> , IEEE Transactions on Computers, Vol. 39, No. 2 (Feb. 1990) (“ Nicol ”)
1012	Arun C. Murthy et al., Apache Hadoop YARN (2014) (“ Murthy ”)
1013	Morris Jette and Mark Grondona, <i>SLURM: Simple Linux Utility for Resource Management</i> , ClusterWorld Conference and Expo (2003) (“ Jette ”)
1014	Eicker et al., <i>The DEEP Project: An Alternative Approach to Heterogeneous Cluster-Computing in the Many-Core Era</i> , Concurrency and Computation: Practice and Experience (2016) (“ Eicker ”)
1015	U.S. Patent App. Pub. No. U.S. 2017/0262319 (“ Newburn ”)
1016	Carsten Clauss et al., <i>Dynamic Process Management with Allocation-Internal Co-Scheduling Towards Interactive Supercomputing</i> , Proceedings of the First Workshop on Co-Scheduling of HPC Applications (2016) (“ Clauss ”)
1018	Samik Raychaudhuri, <i>Introduction to Monte Carlo Simulation</i> , Proceedings of the 2008 Winter Simulation Conference (2008) (“ Raychaudhuri ”)
1019	David W. Bauer, Jr. and Mojdeh Mohtashemi, <i>An Application of Parallel Monte Carlo Modeling for Real-Time Disease Surveillance</i> , Proceedings of the 2008 Winter Simulation Conference (2008) (“ Bauer ”)
1020	List of Challenged Claims
1021	U.S. Patent No. 5,881,227 (“ Brenner ”)
1022	U.S. Patent No. 7,047,299 (“ Curtis ”)
1023	N. Metropolis, <i>The Beginning of the Monte Carlo Method</i> , Los Alamos Science, Special Issue 125-130 (1987) (“ Metropolis ”)

No.	Description
1029	Transcript of Deposition of David R. Kaeli, Ph.D. dated December 9, 2025
1032	Intel Xeon Processor E5 v4 Product Family Datasheet (June 2016)
1033	Intel Xeon Phi Coprocessor x200 Product Family Datasheet (Apr. 2017)
1034	Mellanox InfiniBand FAQ (Dec. 2014)

IV. UNDERSTANDING OF PATENT LAW

7. I incorporate by reference the description of my understanding of patent law presented in my Initial Declaration. *See* EX1003, ¶¶13-22, Appendix A.

8. I further understand that prior art must be enabling in order to invalidate a challenged claim. I understand that a prior art reference is presumed to be enabling. In the context of obviousness, I understand that to render a claim obvious, the prior art, taken as a whole, must enable a skilled artisan to make and use the claimed invention.

9. I understand that enablement relates to whether a disclosure describes an invention in such terms that one skilled in the art can make and use the invention. The standard for determining whether a disclosure is enabling is whether practicing the invention based on the prior art would require undue experimentation by a POSITA. I understand that several factors are often applied to determine whether experimentation is reasonable or undue, including:

- The breadth of the claims;
- The nature of the invention;
- The state of the prior art;
- The level of one of ordinary skill;
- The level of predictability in the art;
- The amount of direction provided by the inventor;
- The existence of working examples; and
- The quantity of experimentation needed to make or use the invention based on the content of the disclosure.

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

V. LEVEL OF ORDINARY SKILL IN THE ART

11. I have reviewed the level of ordinary skill in the art proposed by Dr. Kaeli in his declaration. *See* EX2021, ¶42. For the reasons set forth in my initial Declaration, I was a person of ordinary skill in the art in 2018 according to the level of ordinary skill proposed by Dr. Kaeli.

12. In this Declaration I apply the definition of a POSITA set forth in my Initial Declaration. EX1003, ¶¶31-32. However, the opinions I expressed in my Initial Declaration or the opinions expressed herein would not change if I applied Dr. Kaeli's proposed level of ordinary skill.

VI. THE '442 PATENT

A. Ground 1

1. Claim 1 Is Obvious And Patent Owner's Arguments Fail

a) Lippert-Budenske Teaches Distributing And Redistributing Sub-Tasks To Both Computation Nodes And Booster Nodes

13. Claim 1 requires that, in a given computing iteration, the claimed sub-tasks are distributed (or re-distributed) to both (i) one or more computing nodes and (ii) one or more booster nodes. My initial Declaration explains how this requirement is satisfied by the Lippert-Budenske combination.

14. Budenske repeatedly teaches mapping (and dynamically remapping) different subtasks to different processor types for a given computing iteration. For example, Budenske expressly teaches to “assign each ... subtask to the processors where it is best suited for execution” (EX1005, 390) within a heterogenous system that may contain “up to four different types of processors, and up to a total of 64 processors (of all types combined)” (*id.*, 391). “These processors will comprise the heterogeneous parallel architecture onto which application tasks will be mapped”. *Id.* Noting that “execution time” of subtasks can vary depending on system state, Budenske teaches “matching and scheduling of application subtasks to processors must be performed dynamically at run time.” *Id.*, 390. Budenske thus teaches evaluating whether to remap subtasks from iteration to iteration: “After all subtasks

have completed execution for a given iteration, ... estimate if it is worthwhile to reconfigure the assignment of processing resources to subtasks to reduce execution time of the next iteration.” *Id.*, 391.

15. This dynamic remapping allows Budenske to “fully utilize the architectural flexibility of such a [heterogeneous parallel computing] system.” EX1005, 390. It aims to reassign subtasks to different processor types in order to reduce execution times from iteration to iteration. This purpose would be undermined by—and is wholly inconsistent with—Dr. Kaeli’s theories that all subtasks must be assigned to the same processor type for a given iteration and across iterations. Dr. Kaeli’s reading would leave the flexible architecture grossly underutilized, thus wasting resources and directly contradicting Budenske’s teachings.

16. Budenske’s subtask mapping is implemented by an HC (“Heterogeneous Computing”) Kernel. Budenske emphasizes changing its subtask-to-processor mappings if desirable: “[i]f it is desirable, the HC Kernel will select a new assignment to use for the next execution iteration through the subtasks. If not, the same assignment will continue to be used.” EX1005, 391. To determine possible subtask-to-processor mappings, the system requires information relating to all processor types. *Id.*, 396-97. First, the HC Kernel Algorithm Database, containing the algorithms used to perform subtasks (*id.*, 392), includes implementations of each

algorithm “for each processor type” (*id.*, 396). Because the Algorithm Database also includes the “expected execution time of each algorithm implementation” (*id.*), this allows Budenske to assign a given subtask to any available processor type to minimize the execution time for the subtask. Second, the HC Kernel needs to know “the total number *and type of the processors* in the platform” so that it knows “*how many processors of each type it has available to assign.*” *Id.* (emphasis added). This further confirms that Budenske assigns multiple processor types during each subtask mapping and remapping, when desirable. Finally, “the last item needed” by the HC Kernel to determine subtask mappings is “the time it takes for *each processor in the platform* to communicate with *every other processor in the platform*” to “determine the expected inter-subtask communication times for possible mappings of subtasks onto the platform.” *Id.*, 396-97. Budenske therefore considers communications between subtasks on different processor types in a given computing iteration. Accordingly, Budenske’s HC Kernel considers every processor type available to the platform, as well as the communications of every processor of every type, in determining a resource allocation for a given computing iteration. Budenske therefore teaches both distributing a plurality of subtasks between different processor types in a given computing iteration as well as changing what processor type a given subtask is assigned to in subsequent computing iterations.

b) Dr. Kaeli's Reading Of Budenske Is Incorrect

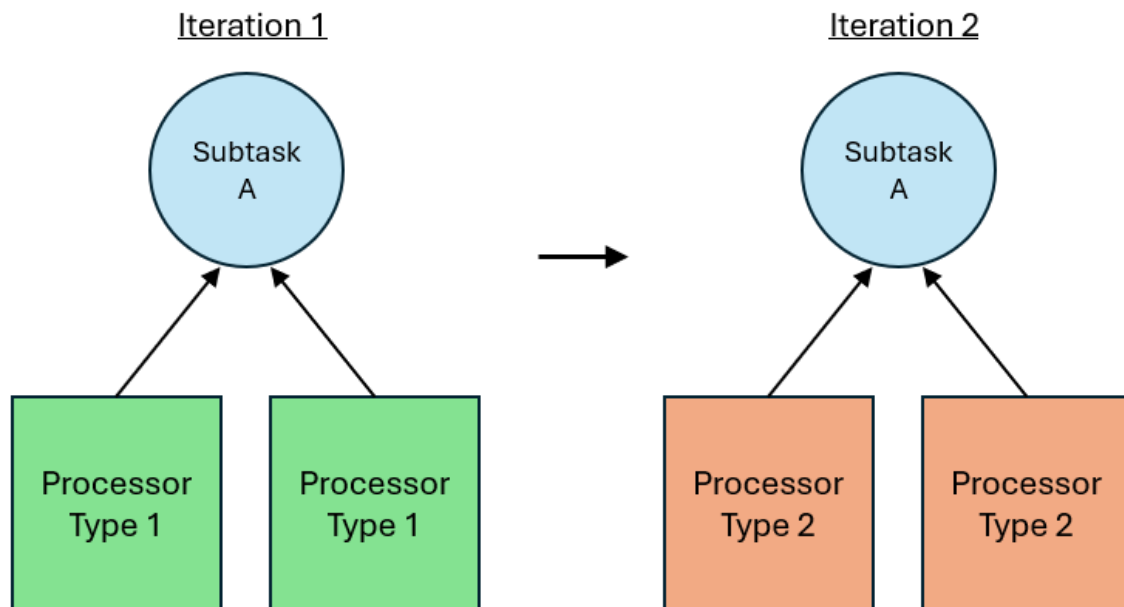
17. Dr. Kaeli asserts that Budenske “fails to teach or render obvious distributing sub-tasks to both computation nodes and booster nodes in a further computing iteration.” EX2021, ¶62. Dr. Kaeli's basis for this position is the following sentence of Budenske: For simplicity, “it is assumed that if an implementation of a given subtask uses multiple processors, all processors will be of the same type.” EX1005, 392 (quoted in EX2021, ¶¶52, 62). Dr. Kaeli's interpretation of this sentence is incorrect as shown by considering the sentence in context of the rest of Budenske. The sentence refers only to a single “implementation of a given subtask,” i.e., an individual subtask. Budenske explains that an implementation of a subtask is constrained to a given computing iteration. *See* EX1005, 402 (explaining that between computing iterations, the HC Kernel determines whether a remapping is desirable and that “this decision is made in real time *after all subtasks' implementations for the current iteration have finished executing and before any subtask implementations begin to execute for the next iteration*”); *id.*, 400 (explaining that where the HC Kernel's genetic algorithm “evaluates a given possible mapping solution (i.e., assignment of resources), it selects the implementation for the subtask that has the smallest expected execution time for the resources assigned to that subtask”). Accordingly, Budenske discloses that an “implementation” of a subtask is limited to the computing iteration in which

it is executed. Further, this sentence of Budenske is permissive and not mandatory, only applying “*if* an implementation of a given subtask uses multiple processors.” *Id.*, 392 (emphasis added). In other words, during a given implementation (i.e., iteration) for a given subtask, that individual subtask *may* be jointly processed by multiple processors of the same type.

18. This understanding was confirmed by Dr. Kaeli in deposition. *See* EX1029, 96:13-24 (“Q. And where it says ‘a particular multiprocessor implementation of a subtask,’ so using that example that you discussed, does that mean that those six SHARC processors are all processing that given subtask? ... A. It means that ... for that particular subtask that uses multiple processors where all the processors are of the same type, ... there will be an implementation of that subtask that will run on the six SHARC processors and ... work collectively.”); *see also id.*, 96:25-97:9. In other words, an implementation of a given subtask may use multiple same-type processors working collectively to more efficiently process that given subtask. But that does not limit other subtasks to the same type of processors or limit subtasks from being assigned to different processors in subsequent iterations.

19. Accordingly, Budenske allows a given subtask to be jointly processed by multiple processors of the same type during a given computing iteration; this neither prevents Budenske from assigning multiple subtasks to different processor types in a given iteration nor prevents Budenske from re-assigning the given subtask

to a different processor type in a subsequent iteration. As an example, the below figure depicts a scenario of a given subtask (Subtask A) being jointly processed by multiple processors of a same first type in a first implementation and multiple processors of the same type (but different from the first type) in a further iteration.



To be clear, while the above example only addresses a “given subtask”, Budenske also allows for other subtasks (e.g., a “Subtask B”) in a given iteration to be executed by a different processor than that used in executing Subtask A. The above is intended to depict the concept of using multiple processors of the same type in an implementation of a given subtask.

20. Subsequent publications addressing Budenske (and sharing co-author, Howard Siegel) confirm this understanding. *See* EX1008, 156 (stating that “[e]ach

of these subtask may allow the use of multiple processors of the same type for efficient execution” but neither limiting all subtasks to the same processor type nor limiting a given subtask to using different processor types in subsequent iterations) (emphasis added); EX1007, 81 (assuming “[f]or simplicity” that processors of the same type will be used “if a data-parallel implementation of a given subtask uses a virtual machine of processors”). As with Budenske, these publications only describe multiple processors of the same type jointly processing a single subtask, but don’t limit all subtasks to a single processor type or limit the given subtask to a single processor type through all computing iterations.

21. Accordingly, the Lippert-Budenske combination renders obvious the claim 1’s limitations to distributing a plurality of subtasks between computation nodes and booster nodes in both a first and a second computing iteration.

2. Lippert-Budenske Enables The Claimed Invention Of The ’442 Patent

a) Budenske’s Teachings

22. Budenske’s disclosures enable a POSITA to practice the heterogeneous computing systems and methods described therein and particularly those relied on for the Lippert-Budenske combination described in my Initial Declaration. The Lippert-Budenske combination relies on Budenske primarily for its disclosures related to mapping subtasks between heterogeneous processors in an initial iteration,

monitoring the execution of the subtasks to collect information relating to the processing, and assigning remapping subtasks between the heterogeneous processors in a subsequent iteration. *See, e.g.*, EX1003, ¶¶56-67. As my initial Declaration explains, “[a] central module of the IOS (HC Kernel or HC Kernel Monitor) (1) establishes an initial mapping of the sub-tasks onto the processing resources of the heterogeneous computing system, (2) monitors the execution of the sub-tasks, and (3) at the end of each iteration, decides how to change the mapping of the sub-tasks onto the hardware based on information about dynamic parameters obtained from processing the sub-tasks in the preceding iteration.” *Id.*, ¶54. The Lippert-Budenske combination also relies on the HC Kernel in combination with Budenske’s Basic Kernel as disclosing an application manager. *See Id.*, ¶¶80-81. Budenske describes the relied-on HC Kernel including, among other things, its components, and functionality, information required to implement it, and how such information is used. *See* EX1005, 396-404. And Budenske details how the HC Kernel interfaces and communicates with the Basic Kernel (i.e., “the low-level operating system”) and how the Basic Kernel operates to execute an application task (i.e., by “access[ing] the implementation code directly from the Algorithm Database using pointers provided with the loading information” provided by the HC Kernel). *Id.*, 388, 402.

23. Dr. Kaeli points to Budenske's disclosure that its "IOS has not been implemented; such an implementation is a major undertaking and outside the scope of this paper, which is the design concepts for the HC Kernel." EX2021, ¶¶70-71 (quoting EX1005, 388-89). However, the quoted "major undertaking" does not refer to Budenske's "theories" in general, but rather to Budenske's IOS and not the HC Kernel which was the focus of the paper. EX1005, 389. While Budenske states that it's proposed intelligent operating system (IOS) "has not been implemented [and] such an implementation is a major undertaking and outside the scope of this paper," I see nothing in Budenske's descriptions of the architecture, structure, and functionality of that IOS to suggest that such an implementation would be outside the skill of a POSITA or would require an undue amount of experimentation to implement.

24. Budenske's disclosures concerning hardware and architecture do not suggest that Budenske is non-enabling, nor do they impose the architectural restriction asserted by Dr. Kaeli. I understand the Dr. Kaeli to argue that certain disclosures in Budenske regarding exemplary architectures allegedly restrict subtasks to execution on a single processor type. In support, Dr. Kaeli cites the following statement: "For example, a Mercury daughter board can be populated with six SHARC processors that physically share a DRAM." EX1005, 392 (cited by EX2021, ¶52). This statement does not support Dr. Kaeli's position for multiple

reasons. First, Budenske expressly presents processors sharing a DRAM as part of an “example,” not as a requirement of the architecture. EX1005, 392. Budenske nowhere requires processors of the same type to share memory (e.g., DRAM), nor does it condition execution or remapping of subtasks on such a configuration. Second, Budenske’s broader architectural description makes clear that processors of different types are interconnected via system buses (e.g., a VME bus), reflecting that data movement between processor types is both expected and supported. The existence of shared memory among processors of a given type is therefore an optimization for intra-type communication, not a limitation on inter-type execution or remapping. Third, a POSITA would not view Budenske’s exemplary disclosure of processors of the same type sharing memory to preclude sub-tasks from being assigned to different processor types in either a single computing iteration or in subsequent iterations. Budenske does not require data to remain resident in a particular memory associated with a given processor type. To the contrary, Budenske explicitly contemplates data and code migration when resources are reconfigured. For example, Budenske explains that “[w]hen making the decision to reconfigure the resources, ... the HC Kernel Monitor must also consider the time required to perform the reconfiguration (e.g., any code and *data movements that the reconfiguration will require*).” EX1005, 403 (emphasis added). This disclosure would inform a POSITA that moving data between memories associated with

different processor types is contemplated as part of the remapping process. The need to account for the cost of such data movement is a performance consideration, not an architectural prohibition. Accordingly, Budenske's exemplary discussion of shared memory does not restrict subtasks to a single processor type and does not undermine the enablement of Budenske's remapping approach.

25. Dr. Kaeli further cites Budenske's disclosure that "[c]ommunications among processors of the same type are assumed to be *symmetric* in the sense that the conflict-free time for any pair of processors to communicate is the same." EX1005, 392 (cited by EX2021, ¶52). This disclosure, however, does not support Dr. Kaeli's position. As an initial matter, the very preceding sentence of Budenske suggests assigning subtasks to multiple processors: "All the processors of all types will have communication paths to one another." EX1005, 392. Budenske then explains that to "connect processors of different types, a VME bus can be used to provide communications among different collections of processors." *Id.* These disclosures make clear that Budenske expressly contemplates and models communications between different processor types.

26. The assumption of symmetric communication applies only to communications among processors of the same type, and serves as a simplifying assumption for estimating execution and communication times within a homogeneous subset of processors. It does not impose a requirement that

communications between different processor types be symmetric, nor does it preclude such communications. To the contrary, as discussed in paragraph 16, above, Budenske's HC Kernel explicitly considers communication times between all processors in the platform, including processors of different types, when determining potential mappings. A POSITA would therefore understand that symmetric communication is not required for Budenske's system to operate or for subtasks to be mapped or remapped across different processor types. Rather, symmetry is an analytical assumption used to simplify performance modeling within a processor-type cluster, i.e. for jointly processing a given individual subtask in a single computing iteration. *See* EX1005, 392 ("Given this and the symmetry property of the inter-processor communications among processors of the same type, the expected execution time of *a particular multiprocessor implementation of a subtask* is independent of which fixed-size subset of the processors of a given type are assigned to execute the subtask.") (emphasis added). Symmetric communication is relevant when multiple processors of the same type jointly execute a single subtask because it provides predictable coordination within that homogeneous group, whereas subtasks executed on other processor types, including those with inter-subtask data dependencies, communicate via the system interconnect, where communication need not be symmetric and is instead treated as a performance cost in scheduling and remapping decisions. Nothing in Budenske requires symmetric

communications as a condition of heterogeneous task assignment. Likewise, the '442 patent does not require symmetric communications between processors, and a POSITA would not view symmetric communication properties as necessary to enable the inventions claimed in the '442 patent.

**b) Kwok's (EX1007) Successful
Implementation Of Budenske**

27. That Budenske was enabled was further confirmed by EX1007, a paper sharing a co-author with Budenske and whose "objective ... is to implement and evaluate a semi-static methodology, called the on-line use of off-line-derived mappings (denoted as On-Off in subsequent sections), whose conceptual structure was proposed in [Budenske]." EX1007, 79. Not only was EX1007 successful in implementing the methodology of Budenske, it found that "the semi-static approach is effective in that it consistently outperformed a fast dynamic mapping heuristic, and yields reasonable performance." EX1007 presents an experiment comparing results of the "on-off" approach first disclosed in Budenske against three other subtask scheduling algorithms for mapping 10 subtasks across 20 computing iterations in a heterogeneous computing system having four different processor types (designated as processor types 0-3). *Id.*, 81-83. Table 4 of EX1007 compares the mappings for the subtasks for iteration 5 of the experiment; the annotated excerpt of

Table 4 below to show the comparison of Budenske's on-off approach to an ECT dynamic scheduling algorithm.

Table 4
Mappings for iteration 5 of Profile A used by the (a) On-Off approach (adopted after iteration 1), (b) ECT approach (adopted after iteration 4), (c) GA on-line approach, and (d) Ideal approach

s_i	Proc type	ρ	$t_u(s_i)$	Comm.	C_{uv}
(a) <i>On-Off approach</i>					
s_0	1	10	11,217	$s_0 \rightarrow s_1$	77
s_1	1	13	10,864	$s_0 \rightarrow s_2$	184
s_2	2	10	13,367	$s_0 \rightarrow s_3$	34
s_3	2	5	12,199	$s_0 \rightarrow s_4$	2551
s_4	3	6	4190	$s_0 \rightarrow s_5$	212
s_5	2	2	18,553	$s_0 \rightarrow s_8$	1523
s_6	0	9	12,175	$s_1 \rightarrow s_7$	64
s_7	2	8	18,346	$s_2 \rightarrow s_9$	3048
s_8	0	15	9355	$s_4 \rightarrow s_6$	1562
s_9	3	10	4738	$s_6 \rightarrow s_7$	2281
				$s_6 \rightarrow s_8$	223
				$s_6 \rightarrow s_9$	7600
(b) <i>ECT approach</i>					
s_0	0	10	8060	$s_0 \rightarrow s_1$	2321
s_1	3	4	9906	$s_0 \rightarrow s_2$	1526
s_2	1	16	18,558	$s_0 \rightarrow s_3$	792
s_3	3	16	4928	$s_0 \rightarrow s_4$	3839
s_4	3	4	4825	$s_0 \rightarrow s_5$	194
s_5	0	14	21,664	$s_0 \rightarrow s_8$	2281
s_6	0	2	41,444	$s_1 \rightarrow s_7$	1551
s_7	0	16	13,440	$s_2 \rightarrow s_9$	176
s_8	2	12	11,798	$s_4 \rightarrow s_6$	1562
s_9	1	14	14,655	$s_6 \rightarrow s_7$	167
				$s_6 \rightarrow s_8$	3040
				$s_6 \rightarrow s_9$	2534

EX1007, 90, Table 4 (annotated and excerpted). The two left-most columns in the table show each subtask number for the 10 processed subtasks (designated s_0 through s_9) and which of the four processor types (0-3) the subtask was mapped to in the computing iteration.

28. These results show that the method of Budenske distributed subtasks between different processor types in a given iteration because each of processor types 0-3 has at least one subtask mapped to it and confirms that “mapping” and “remapping” as disclosed in Budenske means which of multiple potential processor types a given subtask is assigned to in a given computing iteration. EX1007 therefore shows that Budenske enabled its teachings because those teachings were implemented, evaluated, and found to perform well.

29. Dr. Kaeli makes a number of assertions concerning EX1007’s disclosures that are incorrect. First, Dr. Kaeli relies on EX1007’s description of a “fundamental research issue” associated with the implementation of Budenske’s method. EX2021, ¶¶75-78 (quoting EX1007, 80). The described “fundamental research issue” Budenske states needed addressing is one aspect: “how to select representative mappings off-line for on-line use.” EX1007, 80. The very next sentence states that “[t]o solve this problem, a novel dynamic parameter space partitioning and sampling scheme is proposed” and that experimental results “indicate that this semi-static approach is effective.” *Id.* EX1007 itself solves the supposed “fundamental research issue.”

30. Further, Budenske already explained that “[f]or general [heterogeneous computing], deriving an optimal matching and scheduling is intractable (i.e., it is known to be an NP-complete problem that requires exponential execution time to

perform an exhaustive search of the space of possible solutions),” but that this intractability is addressed using a heuristic, “as is common in the heterogeneous computing field.” EX1005, 398; *see also* EX1008, 136 (explaining that “[i]t is well known that such a mapping problem is, in general, NP complete” but that “both static and dynamic heuristics” have been developed to address the problem). Budenske itself explains that “any good heuristic could be employed” with its architecture, including “one that can be executed during run time, such as a levelized-min-time heuristic, or an off-line heuristic, such as a genetic algorithm” (EX1005, 398), and provides details for implementing such heuristics. *Id.*, 398-401.

31. Second, Dr. Kaeli incorrectly asserts that EX1007 discloses only a single “heterogeneity factor” h_{iu} for the equation provided for predicting subtask execution times. EX2021, ¶80 (citing EX1007, 83). But EX1007 makes clear that the h_{iu} indicates the relative speed of the subtask s_i on the type of processor used in virtual machine u , but EX1007 is not limited to virtual machine u as it discloses other virtual machines including “virtual machine v .” EX1007, 83. Figure 6(d) of EX1007 confirms that each processor type has its own heterogeneity factor, presenting heterogeneity factors for each of processor types 0-3 for each of subtasks s_1 - s_9 .

s_i	h_{i0}	h_{i1}	h_{i2}	h_{i3}
s_1	0.4897	0.6815	0.7711	0.7503
s_2	0.2828	0.7129	0.4511	0.2725
s_3	0.2575	0.8511	0.5096	0.8779
s_4	0.6337	0.6921	0.8479	0.3451
s_5	0.8283	0.2745	0.4114	0.2836
s_6	0.7267	0.3124	0.2600	0.3354
s_8	0.3932	0.7026	0.8072	0.8066
s_7	0.5276	0.7990	0.5081	0.7942
s_8	0.5876	0.3863	0.6515	0.6472
s_9	0.8760	0.8794	0.6965	0.2407

(d)

EX1007, 88, Fig. 6; *see also id.*, 88 (“Fig. 6.... (d) heterogeneity factors h_{iu} for the subtask.”). This table unmistakably shows that this implementation of Budenske considers a heterogeneity factor associated with every available processor type (types 0-3) in the platform for each subtask, representing the subtask’s affinity for each processor type. These different affinities can be substantial (e.g., a more than 0.5 difference between the heterogeneity factors for processor types 2 and 3 for subtask 4) or marginal (an approximately 0.01 difference between the factors for types 0 and 3 for subtask 2). And as explained in EX1007, this heterogeneity factor for each processor type is used to model the execution time for a given subtask, which is used to determine the appropriate processor type to map the subtask to. EX1007, 83, Equation (1); *see also id.*, 84 (explaining that for on-line retrieval of

off-line derived mappings, “[i]f the stored pre-determined execution time of the selected mapping, plus the estimated reconfiguration time, is smaller than the execution time that occurred for iteration $i - 1$, a remapping is performed”).

32. Further, EX1007’s disclosure of subtasks having different affinities for different machines does not mean that such subtasks would be assigned to only one machine type. Affinity of certain computing tasks/subtasks to one processor type over another was a well-known concept and simply means that a given computing task will perform better on some machine types relative to others. Affinity for a given processor type does not limit a sub-task to a single processor type but simply means that a processor type will be more efficient in processing a subtask that has affinity for that type. Indeed, the ’442 patent recognizes this very same concept. EX1001, 3:51-53 (“Each of these sub-tasks may be more or less suitable for processing by one or more of the computation nodes or one or more of the boosters.”). One classic example of processor type affinity is matrix multiplication. The classic algorithm for that computation can be run on multiple types of processors, including a GPU, CPU, or DSP, but it will process faster on a special purpose CPU. A POSITA, however, would be aware of circumstances when using different processor type than the ideal type would be preferred. One such circumstance is when the overhead of processing the calculation on the ideal processor would outweigh the efficiency gained in doing so. Another circumstance

is when more processors of a given type are available and they could jointly process a given subtask faster than a smaller number of better-suited processor types.

**c) Additional Enabling
Disclosures In the State Of The Art**

33. Other prior art further enables distributing subtasks between computing nodes and booster nodes in a given computing iteration.

34. At the outset, the '442 patent provides no detail on how a POSITA would distribute sub-tasks between different types of processor. The '442 patent's specific effectively just tells a POSITA to do so, without bothering with further detail of how a POSITA would do so. *See* EX1001, 4:27-29 ("The application manager then performs a reassignment of the sub-tasks for a subsequent iteration of the task."). This suggests that a POSITA was aware of the equipment and techniques are known in the art for doing so. Further, the Lippert-Budenske combination also relies on Lippert for, among other things, its heterogenous computing architecture including computation nodes, booster nodes, and an interface to flexibly couple the nodes. EX1003, ¶59-61. Lippert explicitly discloses processing portions of a task (i.e. subtasks) between computation nodes and booster nodes by enabling a computation node to outsource part of a computation task to a booster node. EX1004, [0003], [0013]-[0014]. Indeed, the '442 patent admits it is a "development of the system described" in Lippert and relies on largely the same architectural

elements including connected computation nodes and booster nodes. EX1001, 1:23-27. The '442 patent does not purport to invent novel hardware and admits that its computation nodes, booster nodes, and interconnect between the nodes may consist of off-the-shelf consumer products. *See id.*, 3:24-36 (describing that such components may comprise “the Intel Xeon E5-2680 processor,” “an Intel Xeon Phi many core processor,” and “a Mellanox InfiniBand ConnectX fabric,” respectively); EX1029, 29:7-17, 30:6-8, 117:1-22. Intel Xeon and Intel Xeon Phi processors and Mellanox InfiniBand interconnect were commercially available products known to a POSITA EX1032; EX1033; EX1034. Accordingly, the '442 patent does not require any novel hardware and a POSITA was aware of known techniques for distributing sub-tasks between different processor types. A POSITA would therefore not expect undue experimentation in applying Budenske’s techniques to this known hardware.

35. Dr. Kaeli cites EX2013 as supporting the following opinions:

The fact that the processors were general purpose meant that there would be many tasks for which they were neither the fastest nor most efficient choice. General purpose CPUs are optimized to benefit from single-thread performance, devoting a significant percent of die space to large caches and branch prediction hardware. In contrast, many accelerators (e.g., GPUs) are optimized for executing a massive number of threads, with a majority of

die space devoted to a large number (thousands) of computing pipelines (over 4000 pipelines in 2018). EX2013. These booster devices (such as graphics processing units, or GPUs) can perform certain tasks far faster or more efficiently than a general-purpose CPU. This is especially true for today's demanding applications that perform the same operation on compound datatypes, such as matrices and vectors. Thus, certain tasks that are far more efficiently executed on a booster can be assigned to the booster rather than the CPU.

EX2021, ¶35. Dr. Kaeli's statement is not entirely factually correct because, for example, a POSITA did not consider CPU's to be optimized to benefit single-thread performance. A POSITA understood many CPUs to include multiple computing cores allowing for effective multi-thread processing. Regardless, the above statement by Dr. Kaeli demonstrates that a POSITA was well aware of CPUs and GPUs, and their respective advantages and disadvantages in processing certain operations, and a POSITA would be capable of implementing Budenske's teachings concerning heterogeneous platforms using CPUs and GPUs.

36. The '442 patent does not purport to invent any novel type of computing task that a POSITA would not find Budenske's disclosures of distributing subtasks between processor types to be applicable to. To the contrary, the only example of an iterative computing job disclosed in the '442 patent is a "Monte-Carlo" based

simulation. EX1001, 3:44-49. Monte-Carlo simulations were first developed in the 1940s and Dr. Kaeli, in deposition, testified to having used “approximately 20 years ago” (EX1029, 110:17-24). *See also* EX1018, 91-99; EX1019, 1029-36; EX1023, 125-30. Accordingly, a POSITA would recognize Budenske’s subtask distribution techniques to full enable distributing the same types of iterative computing subtasks contemplated by the ’442 patent.

37. Finally, Dr. Kaeli present two publications that allegedly illustrate “the types of issues that make it so difficult to map sub-tasks across different types of processors.” EX2021, ¶¶38-39. First, Dr. Kaeli cites to EX2014 for its discussion of a “straggler problem” that can result “lower GPU utilization.” EX2021, ¶38 (quoting EX2014, 179-80). Second, Dr. Kaeli cites EX2015 for alleged disclosure that utilizing two different architectures introduces more complexity than older models. EX2021, ¶39 (citing EX2015). Nothing in these articles or Dr. Kaeli’s description thereof suggests that they would either render Budenske inoperable or prevent the Budenske-Lippert combination from enabling the alleged invention of the ’442 patent. Moreover, both of these articles address problems applying heterogeneous systems to specific machine learning applications requiring high computational performance. *See* EX2014, 179 (“[M]any researchers have struggled to efficiently conduct distributed training on heterogeneous GPU clusters owing to the straggler problem.”); EX2015, 1 (“We observe a continuously increased use of

Deep Learning (DL) ... that requires powerful computing resources with equally increasing performance.”), *id.* 1-2 (distinguishing “*traditional HPC applications*’ (i.e., modeling and simulation sciences) that use iterative methods and rely heavily on a small number of numerical algorithmic classes.”). The ’442 patent does not purport to invent any new type of computing task or application and the only application type explicitly disclosed are conventional Monte-Carlo simulations, i.e. a traditional HPC application. The above articles do not suggest that the Lippert-Budenske combination would not enable implementing the claimed invention for a traditional HPC application having relatively low compute requirements such as a Monte-Carlo simulation. A POSITA would not understand the problems presented in EX2014 and EX2015 to be a major impediment to enabling the claims of the ’442 patent for traditional HPC applications such as Monte-Carlo simulations

38. It is therefore my opinion that the proposed Lippert-Budenske combination enables claim 1 of the ’442 patent.

3. A POSITA Would Have Combined Lippert And Budenske

39. As explained in Section VI.A.1., above, Budenske’s subtasks are not limited to the same type of processor and this therefore would not weigh or teach against combining Budenske and Lippert. Budenske’s application to ATR tasks would also not teach away from combining Budenske with Lippert Again, the claims of the ’442 patent are not limited to any specific kind of iterative computation task

and the specification only specifically discloses conventional Monte-Carlo simulations as an example. And nothing in Budenske suggests that its teachings would only be applicable to ATR tasks and not to other types of iterative computing tasks. In fact, a POSITA reading the disclosures of Budenske would immediately recognize its disclosed systems and methods to be applicable to the processing of other types of iterative computing tasks.

40. EX1007 does not suggest that a POSITA would not have a reasonable expectation of success in combining the references. As explained in paragraph 28, above, the authors of EX1007 explicitly solved the “fundamental research” issue addressed therein. Dr. Kaeli also relies on EX1007’s disclosure that two methods proposed in Budenske for determining representative dynamic parameter values were not “ever evaluated in any way.” EX2021, ¶79 (quoting EX1007, 79). Instead, EX1007 recognizes that in Budenske, “two methods were suggested for determining the representative choices for the dynamic parameter values to use for each static mapping.... *Neither of these methods was ever evaluated in any way.*” EX1007, 79. EX1007 describes one of the proposed methods as a “simplistic approach,” further indicating that a POSITA would have had a reasonable expectation of success in implementing the methods of Budenske. *Id.*

41. Dr. Kaeli asserts that Kambatla reinforces non-obviousness by allegedly describing difficulties in resource usage and distributing tasks. EX2021,

¶113. But this is an issue that Budenske's method specifically addresses. Budenske's HC Kernel considers various items of information to determine the resource usage requirements of a given subtask and the resource capabilities of a given heterogeneous computing system in order to optimally distribute sub-tasks among the various processor types of the system. These include information related to resource usage and resource requirements including (1) the algorithm implementation time for processing a sub-task in a given computing iteration; (2) the number and type of all processors in the platform; and (3) the communication times between all processors in the platform. EX1005, 396. And Budenske considers the dynamic parameters of subtasks in real-time to select an appropriate mapping for a given computing iteration. *Id.*, 402. Thus, a POSITA would be motivated to implement Budenske to address these very issues of estimating resource usage and distributing subtasks to different processor types effectively.

4. Budenske Generates In Real-Time A Further Distribution Of Sub-Tasks Using Information Relating To The Processing Of The Plurality Of Sub-Tasks

42. Budenske includes an off-line component of preparing potential candidate mappings prior to processing sub-tasks. EX1005, 396 ("The HC Kernel MDDG Generator is the component of the off-line IOS ... that is responsible for mapping each ODDG onto the heterogenous parallel hardware platform creating a corresponding MDDG."). This minimizes processing run-time by calculating

potential mappings of subtasks to processor types prior to subtask execution. The HC Kernel MDDG Generator composes such potential mappings (MDDGs) for multiple potential scenarios corresponding to different dynamic parameter values that might be encountered during processing. EX1005, 398.

43. During sub-task execution, Budenske's HC Kernel monitors the sub-task's dynamic parameter values (i.e., information relating to the processing of the plurality of sub-tasks) and, based on those observed values, selects an appropriate mapping from among the pre-calculated candidate mappings, and "*instantiates*" (*generates*) that mapping to effect a further distribution of subtasks between different processor types in the next computing iteration. EX1003, ¶¶77-78; *see also* EX1005, 395 (explaining that "as the application is executing, ... at the end of each iteration ... make a heuristically-based decision whether to ... select and *instantiate* a new mapping (for the next iteration)"); *id.*, 398 ("[I]t is possible to use off-line precomputed mappings to *reconfigure resources in real time.*"); *id.*, 403 ("[T]he HC Kernel Monitor can use the actual values of dynamic parameters at the end of each execution iteration of the application to make remapping decisions and select new mappings derived by a time-consuming off-line heuristic."). Moreover, Budenske makes clear that instantiating a selected mapping includes reconfiguring the system as necessary to distribute subtasks to processing resources in accordance with that mapping. *Id.* (explaining that the HC Kernel considers "e.g., any code and data

movements that the reconfiguration will require”). Thus, a POSITA would understand that “instantiating” a mapping in Budenske means generating and implementing the selected mapping in the running system, including assigning subtasks to processors and performing any required code or data movement. From a POSITA’s perspective, particularly in software and systems engineering, to instantiate a mapping would naturally be understood to mean to create and put into effect the selected configuration in the executing system, as opposed to merely identifying it in theory. Budenske’s explicit discussion of reconfiguration costs (including code and data movement) confirms this understanding and provides intrinsic support for that interpretation. Accordingly, in Budenske, information relating to the processing of subtasks is used to generate and implement a distribution of subtasks in a further computing iteration.

44. Claim 1 of the ’442 patent requires “generating ... a further distribution of the plurality of sub-tasks” and a POSITA would understand that generating such a distribution includes the step instantiating that distribution in the computing system; that is, putting the distribution into effect by assigning the subtasks to processing resources for the subsequent computing iteration. Although calculating a candidate mapping may be part of generating a distribution, generation at least includes implementing a selected distribution in the system so as to effectuate the distribution of subtasks for the next computing iteration. Accordingly, Budenske’s

disclosure that the HC Kernel “select[s] and instantiate[s] a new mapping (for the next iteration)” falls squarely within the scope of generating a further distribution, as instantiation is the act by which the selected distribution is put into effect in the system.

5. Budenske’s Sub-Tasks Are Information Relating To The Processing Of The Plurality Of Subtasks

45. The ’442 patent demonstrates that a POSITA would not understand the claimed “information” to be limited to “the information from the daemons concerning the processing of the sub-tasks.” For example, claim 5 further narrows claim 1 to claim that “daemons operate in the plurality of computation nodes and the plurality of booster nodes to generate the information.” EX1001, 6:12-14. This suggests that the claimed “information” is broader than just such information generated by daemons.

46. Further, the ’442 patent’s prosecution history confirms that the claimed “information” includes at least the scalability of a subtask. EX1002, 174-75 (“[A]mended claims 1 and 9 allow for redistribution based on information relating to the processing of the plurality of sub-tasks, *including adjusting the distribution of sub-tasks between computation nodes and boosters based on a scalability of the sub-task.*”).

47. A POSITA would understand the dynamic parameters disclosed in Budenske to be analogous to and used in the same way as the scalability of a subtask utilized in the '442 patent. The '442 patent explains that scalability is a property of the processed subtask and may be determined prior to a computing iteration either by an operator or automatically by the application manager. EX1001, 4:38-43. Likewise, the dynamic parameters of Budenske reflect properties associated with the sub-tasks. EX1005, 393, (explaining that the dynamic parameters “change during run time” and “have the most impact on the execution time of the subtasks in the application.”), *id.* (“Each of these sets of Application and Input Data Characteristics can be divided into static and dynamic parameters.”); *id.*, 403 (“[T]he values of dynamic parameters depend on the input data.”); *id.*, Abstract (describing “perform[ing] a remapping of the application onto the platform based on information generated from input data”). Therefore, a POSITA would understand information relating to the processing of the plurality of subtasks to include the dynamic parameters disclosed in Budenske.

B. Claim 3

48. The '442 patent nowhere describes its resource manager as altering an assignment of subtasks to computation nodes and booster nodes during the computing of a computation task. The '442 only describes that the resource manager may determine such assignment “for the first iteration.” EX1001, 2:19-22. Nowhere

does the '442 patent describe the resource manager as altering such an assignment. Instead, the '442 patent consistently describes the resource manager as altering the assignment of computation nodes and booster nodes to one another:

- “In a further aspect of the invention, the resource manager, based on the information dynamically alters assignment of the computation nodes and booster nodes to each other during the computation of the computation task.” EX1001, 2:26-29.
- “[W]herein the system comprises a resource manager for assigning booster nodes and computation nodes to each other for the computing of the tasks.” *Id.*, 2:41-44.
- “In still a further aspect of the invention, the resource manager receives the information such that the resource manager dynamically alters assignment of the computation nodes and booster nodes to each other during the computation of the computation task.” *Id.*, 2:51-55.
- “The resource manager can also reassign computation nodes and boosters to each other, in addition to the adjusted distribution of tasks.” *Id.*, 4:13-15.
- “This task is compiled and the compiled code is executed. In the execution, a task is analyzed by the application manager and sub-tasks of the task are divided into computation node suitable sub-tasks and

booster suitable sub-tasks and this information is passed to the resource manager for assignment of boosters to computation nodes.” *Id.*, 4:18-23.

49. As for re-assignment of subtasks to computation and booster nodes in subsequent computing iterations, the ’442 patent consistently describes this functionality performed by the application manager and not the resource manager. Indeed, this functionality is claimed in claim 2, from which claim 3 depends.

- “The information provided allows an application manager to determine if the distribution of sub-tasks between computation nodes and booster nodes can be adapted or improved upon for a further computing iteration.” EX1001, 2:15-20.
- “[T]he application manager being arranged to receive information from daemons operating in the computation nodes and booster nodes to update a distribution of the sub-tasks between the computation nodes and the booster nodes between a first computing iteration and a further computing iteration.” *Id.*, 2:45-50.
- “The application manager then performs a reassignment of the sub-tasks for a subsequent iteration of the task and passes this updated assignment information to the resource manager which may also adjust the assignment of boosters to nodes accordingly.” *Id.*, 4:27-31.

- “Accordingly, the application manager is arranged using the information provided by the daemons operating in the computation nodes and the boosters to adjust groupings of sub-tasks for assignment to the computation nodes and boosters.” *Id.*, 4:59-63.

50. Consistent with the above disclosures, a POSITA would understand claim 3’s limitation of the “resource manager dynamically alter[ing] further distribution during the computing of the computation task” to include at least the consistently described functionality of the resource manager assigning computation nodes and booster nodes to each other.

51. Even if claim 3’s limitation of “resource manager dynamically alter[ing] further distribution during the computing of the computation task” refers to the “further distribution of the plurality of sub-tasks” recited in claim 1, claim 3 would still be obvious in light of Lippert-Budenske. As I explained in my Initial Declaration, the application manager of Lippert-Budenske determines a new distribution of subtasks and provides this new distribution to the RM of Lippert-Budenske such that the RM dynamically alters the distribution of computation nodes and booster nodes to each other “to implement the new mapping.” EX1003, ¶¶87-88; *id.*, ¶¶80-81, 83-85 (describing the AM and RM of Lippert-Budenske). A POSITA would have been further motivated to implement the RM to dynamically alter the further distribution of the plurality of sub-tasks in order to match how the

RM dynamically altered assignment of computation nodes and booster nodes to each other, because doing so would “fully utilize the architectural flexibility” (EX1005, 390; EX1004, [0015]), and “provide fault tolerance in case of accelerator failures” (EX1004, [0003]). For example, if there is a “lack of resources” (EX1004, [0003]) due to other jobs being processed and/or booster failure, the RM may be limited in how it can dynamically alter the computation node-booster assignment and may need to adapt the sub-task assignment according to what resources are available for the next iteration. The POSITA would have reasonably expected to succeed in implementing the RM in this way because the RM already “handles an initial mapping of sub-tasks” (EX1003, ¶82), and for the same reasons discussed in my Initial Declaration for combining the teachings of Lippert and Budenske (EX1003, ¶¶ 59-63).

C. Claim 5

**1. Lippert-Budenske Renders Obvious
Computation Node And Booster Node Daemons**

52. Both Lippert and Budenske disclose collecting processing information, including detecting computing capacities of boosters and dynamic parameters. EX1003, ¶90. This suggests the existence of logic for performing such information collection and a POSITA would understand the need to include a logic for performing such detection and would implement such logic as a background process

for performing that specific task (i.e., a daemon). *Id.*; *see also* EX2021, ¶90 (Dr. Kaeli stating that a daemon “is sometimes used to refer to as [sic] a background process.”). Lippert explicitly discloses daemons, explaining that its resource manager may be implemented as a daemon. EX1004, [0014]; *see also* EX1003, ¶57. A POSITA would be motivated to implement such background processes at individual computation and booster nodes, i.e. to “facilitate inter-element communication while providing communication authentication beneficially, and to enable independent control of task execution and reporting of task processing at each node.” EX1003, ¶90.

D. Claim 9

53. The opinions expressed above in Sections VI.A-C apply equally to claim 9.

VII. GROUND 2

54. The opinions expressed above for Ground 1 concerning the Lippert-Budenske combination apply equally to Ground 2 concerning the Lippert-Budenske-Kambatla combination.

A. Claim 3

55. The proposed Lippert-Budenske-Kambatla combination relies on the computing hardware elements (computation nodes, booster nodes, and communication infrastructure) of Lippert. Lippert-Budenske-Kambatla relies on the

same resource manager functionality of Lippert-Budenske of dynamically altering further distribution of computing nodes and booster nodes. EX1003, ¶126. A POSITA would be motivated to include such a module as part of a central resource manager as taught by Kambatla to “achiev[e] the benefits of Kambatla’s resource manager and the dynamic assignment of Lippert-Budenske’s resource manager.” *Id.*, ¶¶125-26.

B. Claim 5

56. Dr. Kaeli asserts that Lippert-Budenske-Kambatla fails to disclose daemons in computation and booster nodes. Dr. Kaeli argues that Kambatla does not discuss the concept of heterogeneous computing for processing tasks. EX2021, ¶¶102-04. First, that assertion is incorrect because Kambatla discloses that its “worker nodes may have different hardware” and therefore contemplates platforms comprising nodes with differing characteristics and capabilities. EX1006, [0059], [0071]. Further Kambatla discloses that the aggressiveness of its scheduling can be controlled through variable parameters designated as T_{alloc} and T_{preempt} . Kambatla discloses that these parameters may be “specific to a worker node given the characteristics and/or capabilities of the worker node” and that the parameters can “be generalized to a vector, with dimensions representing *different resource types*.” These disclosures demonstrate that Kambatla contemplates scheduling decisions based on worker nodes comprising different resource types. Further, a POSITA

would understand the teachings of Kambatla to be applicable to Lippert-Budenske even if it did not disclose a heterogeneous computing system. Each worker node disclosed in Kambatla (i.e. each processing node) includes a daemon that monitors local conditions and reports information relevant to scheduling decisions. A POSITA implementing the teachings of Kambatla into Lippert-Budenske would implement such a daemon at each processing node of the combined system, regardless of whether such nodes are booster or computation nodes. As explained above, the '442 patent does not disclose any novel node architecture, instead allowing computation and booster nodes to comprise off-the shelf commercial processors. Nothing in Lippert or Budenske would prevent a POSITA from implementing a background software process on such nodes. Indeed, implementing a background process (i.e, a daemon) to into the computation and booster nodes taught by Lippert to collect and report dynamic parameter values as taught by Budenske, would be a straightforward and routine task for a POSITA. Such a daemon could be implemented in software, for example, as a background loop or event-driven process that periodically monitors resource usage or task execution characteristics and reports that information to a scheduler or controller. Accordingly, the absence of an explicit daemon label for nodes in Lippert or Budenske does not preclude, and would not discourage, a POSITA from implementing daemons in the combined system.

57. Kambatla complements Lippert-Budenske through providing techniques for optimally scheduling tasks across the processing resources. EX1006, [0004], [0017]. A POSITA would not view such scheduling techniques to be inapplicable to heterogeneous computing systems. Further, Kambatla and Lippert-Budenske share principles addressing allocating cluster computing resources to jobs that typically include multiple tasks and the problem of resource fragmentation and severe under-utilization of computing resources in a cluster. EX1003, ¶¶116-20. These principles apply to the heterogeneous computing platforms taught by Lippert and Budenske regardless of whether Kambatla teaches a platform including different processor types.

C. Claim 9

58. The opinions presented above for claim 1 of Ground 2 apply equally for claim 9 of Ground 2.

D. A POSITA Would Have Been Motivated To Combine Kambatla With Lippert-Budenske

59. As explained in paragraph 55, above, Kambatla teaches processors of different types and a POSITA would consider Kambatla's teachings to be relevant to the heterogenous platforms of Lippert and Budenske and would therefore combine those teachings with Lippert-Budenske with a reasonable expectation of success.

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

Dated: December 22, 2025



Darrell D. E. Long, Ph.D.