

Microsoft Corporation

v.

ParTec AG

IPR2025-00318 | U.S. Patent No. 11,537,442

Demonstratives of Petitioner Microsoft

April 14, 2026

Klarquist

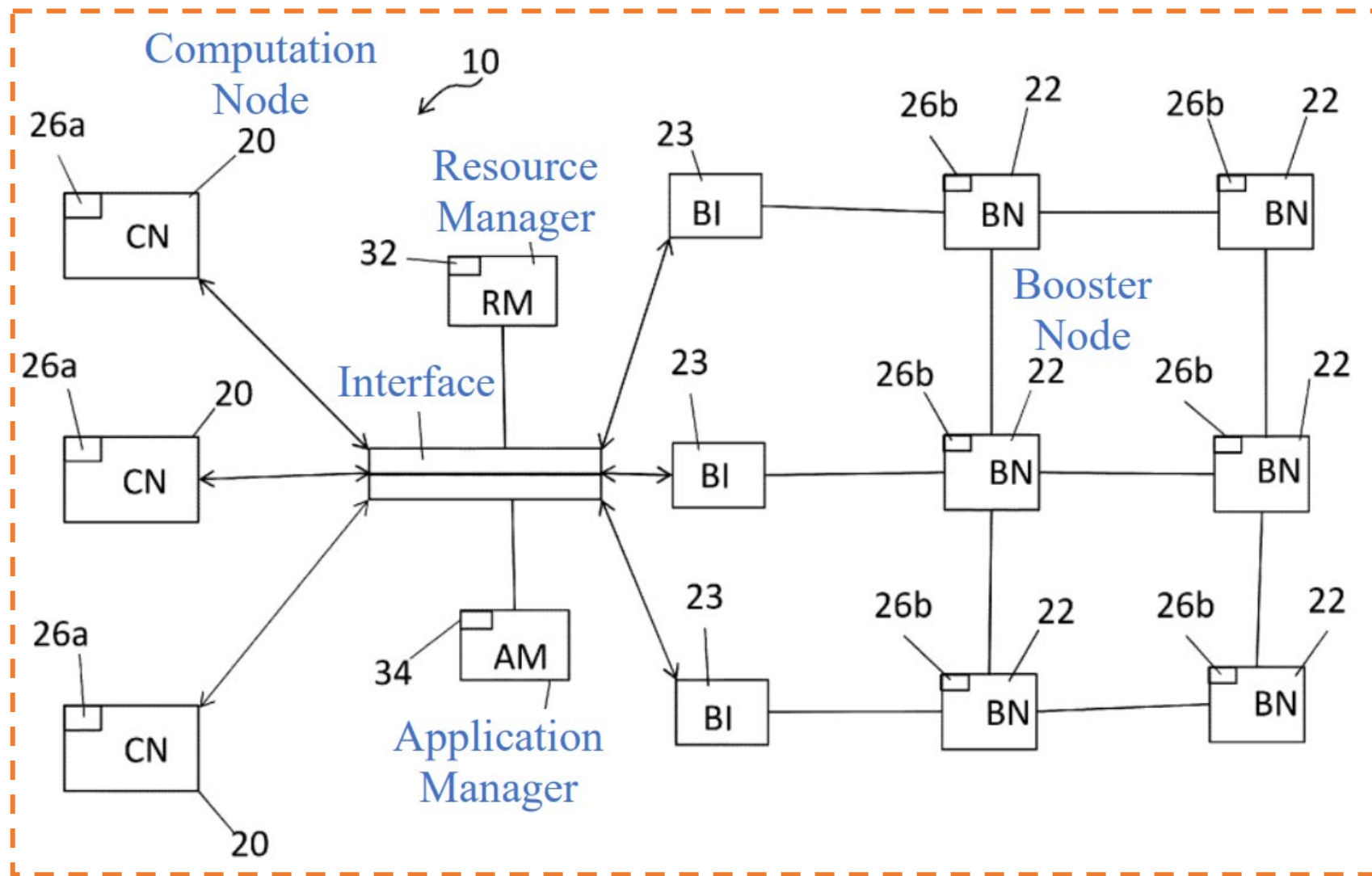
DEMONSTRATIVE EXHIBIT – NOT EVIDENCE

Grounds

	Reference(s)	Basis	Claims
Ground 1	Lippert (EX1004) and Budenske (EX1005)	35 U.S.C. § 103	1-10
Ground 2	Lippert, Budenske, and Kambatla (EX1006)	35 U.S.C. § 103	2-5, 8-10

Petition (Paper 1), 2

'442 Patent



EX1001, FIG. 1
(cited and annotated by
Petition, 5)

Claim 1

[1.1]	A method of operating a heterogeneous computing system comprising a plurality of computation nodes and a plurality of booster nodes, at least one of the plurality of computation nodes and a plurality of booster nodes being arranged to compute a computation task, the computation task comprising a plurality of sub-tasks, the method comprising:
[1.2]	in a first computing iteration, assigning and processing the plurality of subtasks by at least a portion of the plurality of computation nodes and at least a portion of the plurality of booster nodes in a first distribution; and
[1.3]	generating, using information relating to the processing of the plurality of sub-tasks by at least the portion of the plurality of computation nodes and at least the portion of the plurality of booster nodes, a further distribution of the plurality of sub-tasks between the plurality of computation nodes and the plurality of booster nodes for processing thereby in a further computing iteration.

sole disputed
claim 1 element

Petition, 7-8; EX1001, Claim 1

Ground 1 Combination: Lippert-Budenske

Budenske

- “set of subtasks,” repeatedly executed, one iteration after another
- dynamic parameters (e.g., no. of detected objects)
- after each iteration, reconfigure the “assignment of processing resources to subtasks”

After each execution iteration through the set of subtasks, the values of certain dynamic parameters of the application may change, such as the number of objects detected in the current frame of a real-time image stream being processed. It is expected that the values of these parameters will change slowly. After all subtasks have completed execution for a given iteration through the DDG, and before the next iteration begins, the latest values of these dynamic parameters will be reported to the on-line HC Kernel. The HC Kernel will use the most recent values of such dynamic parameters to estimate if it is worthwhile to reconfigure the assignment of processing resources to subtasks to reduce execution time of the next iteration. If 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 (cited by Petition, 1, 14, 17, 19-21, 26, 28, 30-32, 34, 38, 40)

Budenske

For the initial iteration through the set of subtasks, the IOS will use information about the processing environment in its selection of algorithms for the subtasks, and their associated implementations. As part of this, the IOS will decide how to assign resources (e.g., processors) to the subtasks.

EX1005, 391
(cited by Petition, 1, 14, 17, 19-21,
26, 28, 30-32, 34, 38, 40)

Budenske

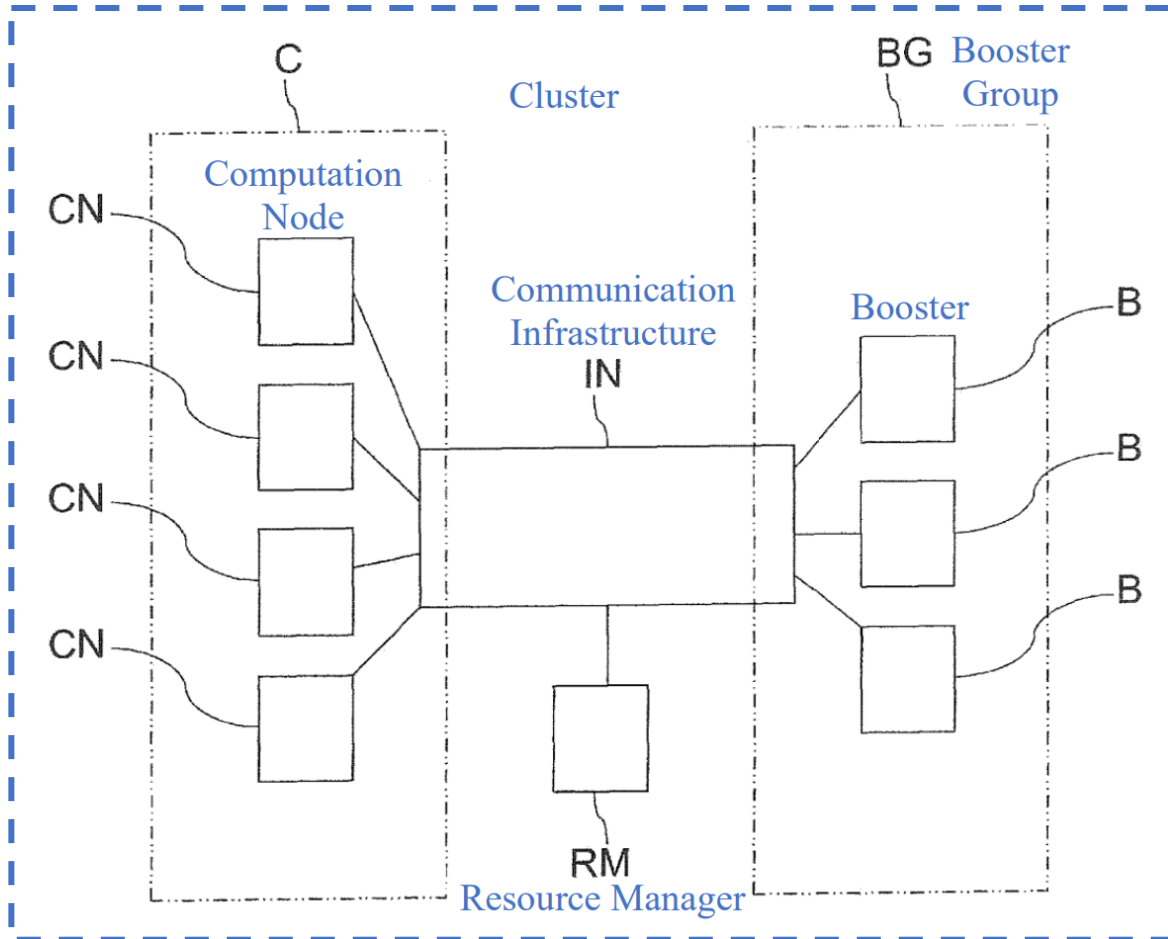
data dependent). Consider the intended application domain and hardware platforms. As an example, assume there are four types of processors, with 16 processors of each type, and assume the application consists of 50 subtasks. Then, an exhaustive search of all possible assignments of subtasks to machines will take time proportional to $(4 \times 16)^{50}$. Thus, a heuristic is used, as is common in the heterogeneous computing field [22], [23].

EX1005, 398 (cited by Petition, 31-32; Reply (Paper 19), 15)

the IOS components are summarized in Figures 1 to 3. The HC Kernel ideas introduced here can also be used for other application domains and classes of hardware platforms whose characteristics are similar to those of the applications and platforms considered here.

EX1005, 404 (cited by Petition, 22)

Lippert



EX1004, FIG. 2 (cited and annotated by Petition, 12)

[0029] Computation nodes typically apply processors comprising an extensive control unit as several computation tasks have to be processed simultaneously. Processors being applied in boosters typically comprise an extensive arithmetic logic unit and a simple control structure when being compared to computation nodes processors. For instance SIMD, also refer to as single instruction multiple data computers, may find application in boosters. Hence, processors being applied in computation nodes differ in their processor design compared to processors being applied in boosters.

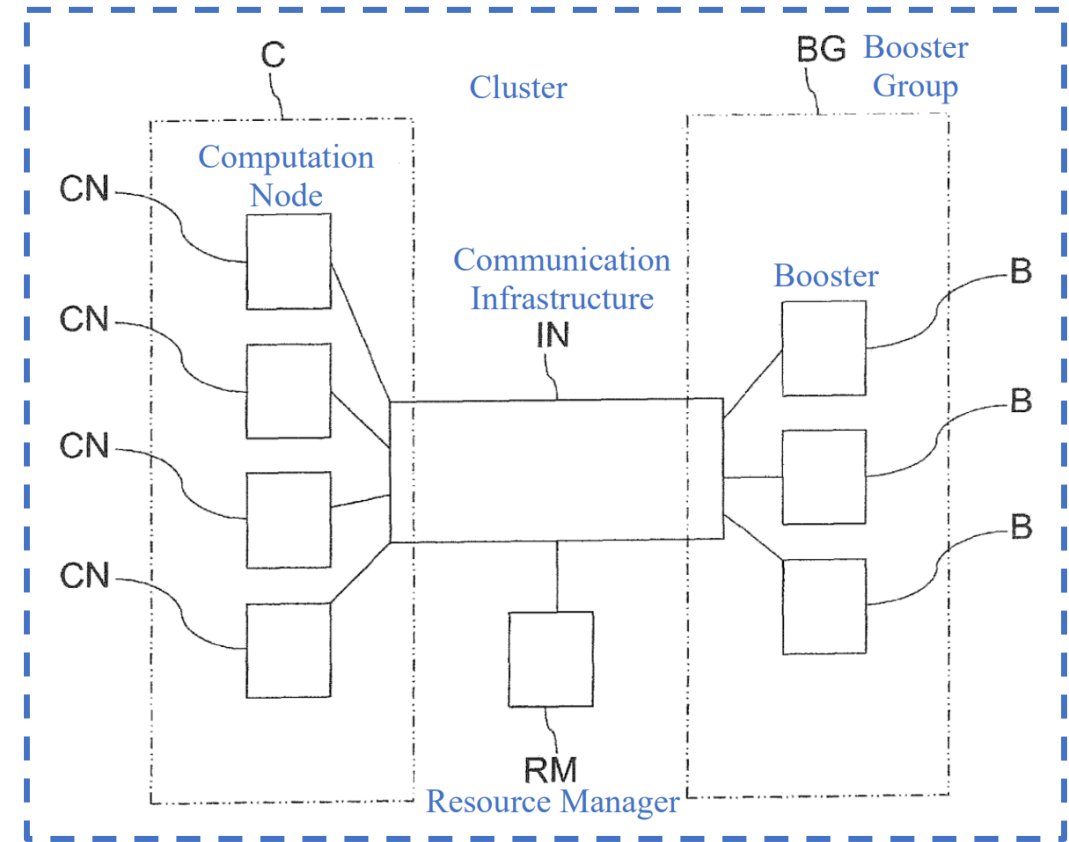
EX1004, [0029] (cited by Petition, 12, 18, 24)

Ground 1 Combination: Lippert and Budenske

The combined Lippert-Budenske system included:

A computer cluster, as taught by Lippert, including:

EX1003, ¶57
(cited by Petition, 15-16)



EX1004, FIG. 2

(cited and annotated by Petition, 12)

Ground 1 Combination: Lippert and Budenske

The combined Lippert-Budenske system included:

- A resource manager module implemented as a daemon, as taught by Lippert (*e.g.*, EX1004, [0014]), arranged to:
 - Assign booster nodes to computation nodes for processing a computation task, as taught by Lippert (*e.g.*, EX1004, [0014]);
 - Assign, according to an initial mapping, sub-tasks of the computation task to the nodes for processing in a first iteration of the computation task, as taught by Budenske (*e.g.*, EX1005, 401-402); and
 - Dynamically assign, during the processing of the computation task, computation nodes, and booster nodes to each other, as taught by Lippert (*e.g.*, EX1004, [0013]);

EX1003, ¶57 (cited by Petition, 15-16)

Ground 1 Combination: Lippert and Budenske

The combined Lippert-Budenske system included:

- An application manager module, as taught by Budenske (*e.g.*, EX1005, 388, 402), arranged to:
 - Deploy the sub-tasks to the computation nodes and the booster nodes, as assigned for a current iteration, for execution of the sub-tasks by the nodes in the current iteration, as taught by Budenske (*e.g.*, EX1005, 402);
 - Monitor execution of sub-tasks to collect information relating to the processing of the sub-tasks in the current iteration, as taught by Budenske (*e.g.*, EX1005, 402); and
 - Assign, according to a new mapping obtained using information related to the processing of the sub-tasks, the sub-tasks to the nodes for a subsequent iteration of the computation task, as taught by Budenske (*e.g.*, EX1005, 401-402).

EX1003, ¶57 (cited by Petition, 15-16)

Claim 1 Issues

Claim 1 Issues

Whether the Lippert-Budenske combination:

1. assigns subtasks to different processor types, in both a first and second distribution, where the second distribution differs from the first

E.g., Petition, 27-32; POR (Paper 16), 23-40; Reply, 1-5

2. satisfies “**generating**, using information relating to the processing ... a further distribution”

E.g., Petition, 29-32; POR, 46-47; Reply, 15-16

3. satisfies “generating, using **information relating to the processing ...** a further distribution”

E.g., Petition, 29-32; POR, 47-50; Reply, 16-18

Claim 1, Issue 1 – assign subtasks to different processor types

data dependent). Consider the intended application domain and hardware platforms. As an example, assume there are four types of processors, with 16 processors of each type, and assume the application consists of 50 subtasks. Then, an exhaustive search of all possible assignments of subtasks to machines will take time proportional to $(4 \times 16)^{50}$. Thus, a heuristic is used, as is common in the heterogeneous computing field [22], [23].

EX1005, 398 (cited by Petition, 31-32; Reply (Paper 19), 15)

creating a corresponding MDDG. Three items of information about the hardware platform will be needed by the HC Kernel MDDG Generator: (P1) algorithm implementation execution time, (P2) number of each type of processor available, and (P3) inter-processor communication time. The HC Kernel MDDG Generator uses this information when it applies the heuristic for determining an effective assignment of subtask implementations to processors.

EX1005, 396 (cited by
Petition, 28-29, 38; Reply, 3)

Claim 1, Issue 1 – dynamically assign each subtask to “best suited” processor

Budenske assigns each subtask to the processor(s) “best suited for execution”:

development of a high-level operating system that can fully utilize the architectural flexibility of such a system. Such a high-level operating system must be able to assign each ATR application subtask to the processors where it is best suited for execution. Often, subtasks can execute concurrently, sharing resources. Because the execution time of application subtasks in an ATR system is highly input-data dependent (e.g., number of currently located objects), this matching and scheduling of application subtasks to processors must be performed dynamically at run time.

EX1005, 390

(cited by Petition, 17-20, 32, 38-39; Reply, 2-3)

Budenske: a given subtask may use multiple processors of the same type

For simplicity, it is assumed that if an implementation of a given subtask uses multiple processors, all processors will be of the same type. 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.

EX1005, 392
(cited by Petition, 29, 51;
Reply, 3-4, 9-10)

Dr. Kaeli: Budenske distributes subtasks to different processor types

8 Q. (BY MR. CLAWSON) Does Budenske describe being
9 able to distribute different subtasks to different
10 processor types in the same computing iteration?

11 MR. LIMBACHER: Objection, form.

12 A. So Budenske describes that a portion of the
13 subtasks can be mapped to one processor type and a
14 portion of the subtasks to another processor type, then
15 based on the input, the actual image for the ATR
16 application, make a change in that allocation of -- of
17 which -- which of the subtasks are mapped to which
18 computing type. And all of that is actually done
19 offline, not online, as is done in the '442.

EX1029, 102:8-19
(cited by Reply, 3-4;
Sur-Reply (Paper 21), 6)

Claim 1, Issue 2 – “generating”

[1.3] **generating**, using information relating to the processing of the plurality of sub-tasks by at least the portion of the plurality of computation nodes and at least the portion of the plurality of booster nodes, **a further distribution of the plurality of sub-tasks** between the plurality of computation nodes and the plurality of booster nodes for processing thereby in a further computing iteration.

Petition, 29 (quoting EX1001, 5:37-43)

Budenske “instantiates” new mappings:

mapping. Then, as the application is executing, **the HC Kernel Monitor monitors the run-time values of the dynamic parameters at the end of each iteration** through the underlying DDG to make a **heuristically-based decision** whether to continue with the current mapping, or to **select and instantiate a new mapping (for the next iteration) from among the entries of the relevant MDDG Table** (which were determined off-line). Thus, the off-line processing provides a set of predetermined mappings that the on-line processing can index in real time.

EX1005, 395 (cited by Petition, 29-30; Reply, 15)

Claim 1, Issue 2 – “generating”

Budenske also teaches determining each mapping “during run time” (and then instantiating that new mapping):

The structure of the HC Kernel MDDG Generator is such that any good heuristic could be employed. The heuristic that is used could be one that can be executed during run time, such as a levelized-min-time heuristic (e.g., [16]), or an off-line heuristic, such as a genetic algorithm (e.g., [20], [24], [28]). In general, an off-line heuristic can find a better mapping than an on-line run-time heuristic because its execution time can be orders of

EX1005, 398 (cited by Petition, 31-32; Reply, 15)

Claim 1, Issue 3 – “information relating to the processing”

[1.3] generating, using information relating to the processing of the plurality of sub-tasks by at least the portion of the plurality of computation nodes and at least the portion of the plurality of booster nodes, a further distribution of the plurality of sub-tasks between the plurality of computation nodes and the plurality of booster nodes for processing thereby in a further computing iteration.

Petition, 29 (quoting EX1001, 5:37-43)

Budenske uses the values of its dynamic parameters to instantiate a new mapping of subtasks:

mapping. Then, as the application is exccuting, the HC Kernel Monitor monitors the run-time values of the dynamic parameters at the end of each iteration through the underlying DDG to make a heuristically-based decision whether to continue with the current mapping, or to select and instantiate a new mapping (for the next iteration) from among the entries of the relevant MDDG Table (which were determined off-line). Thus,

EX1005, 395 (cited by Petition, 29-30; Reply, 15)

Claim 1, Issue 3 – “information relating to the processing”

Budenske’s dynamic parameters are information relating to the processing of the plurality of subtasks:

Consider the following hypothetical example. The task is the tracking of vehicles on the ground, and the Application Developer has chosen four dynamic parameters: (1) number of objects of interest being tracked in the last image frame, (2) average size of the objects currently being tracked (in terms of percent of the total number of pixels in the image), (3) whether the lighting conditions in the last image frame required requantization for better contrast for display, and (4) whether there was cloud cover detected in the last image frame that required special processing. The representative choices specified for the dynamic parameters are: (1) 0, 1, 2, 4, 8, 12, (2) 5%, 10%, 15%, 20%, 40%, (3) yes or no, and (4) yes or no. Thus, $C_0 = 6$, $C_1 = 5$, $C_2 = 2$, $C_3 = 2$, and $S = 120$.

EX1005, 393 (cited by Petition, 19-20, 30-31; Reply, 17)

Claim 1, Issue 3 – “information relating to the processing”

Certain dynamic parameters will have “the most impact on the execution time”:

tion Developer. For the dynamic characteristics, the Application Developer is expected to define a small set of dynamic parameters that will have the most impact on the execution time of the subtasks in the application. It is the job of the *Scenario Generator* component

EX1005, 393 (cited by Petition, 19-20, 30-31; Reply, 17)

Lippert-Budenske
Is Enabling

Lippert-Budenske Enables Distributing Subtasks to Both Processor Types

Budenske teaches use of both online and offline heuristics for determining the mapping:

The structure of the HC Kernel MDDG Generator is such that any good heuristic could be employed. The heuristic that is used could be one that can be executed during run time, such as a levelized-min-time heuristic (e.g., [16]), or an off-line heuristic, such as a genetic algorithm (e.g., [20], [24], [28]). In general, an off-line heuristic can find a better mapping than an on-line run-time heuristic because its execution time can be orders of

EX1005, 398

(cited by Petition, 31-32; Reply, 15)

'442 Patent Provides No Detail On Sub-Task Distribution

PO alleges:

(iii) *Budenske Does Not Enable Distributing Sub-tasks to Both Computation Nodes and Booster Nodes In a Computing Iteration*

POR, 30

The '442 patent:

the status of the nodes. 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.

EX1001, 4:27-31 (cited by Reply, 11)

Prior Art References Not Considered In Isolation For Enablement

“To render a claim obvious, the prior art, taken as a whole,
must enable a skilled artisan to make and use the claimed invention.”

Raytheon Techs. Corp. v. Gen. Elec. Co., 993 F.3d 1374, 1380 (Fed. Cir. 2021)
(cited by Reply, 5)



Long Explained Budenske's Mapping Heuristics

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 leveled-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.

EX1030, ¶ 30 (cited by Reply, 9)

Claim 3

Claim 3: “further distribution”

[3.2] wherein the application manager receives the information and processes the information as input to the resource manager such that the resource manager dynamically alters further distribution during the computing of the computation task.

Petition, 39 (quoting EX1001, 6:1-5)

The '442 patent describes the resource manager altering assignment of computation nodes to booster nodes:

task and further parameters. The application manager processes the information as input to the resource manager such that the resource manager dynamically alters further distributions during the computing of the computation task.

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, 2:22-29
(cited by Reply, 19)

Claim 3: “further distribution”

Lippert discloses a resource manager that dynamically alters assignment of computation nodes and boosters during the processing of the computation task:

module or resource manager node, may be provided. The resource manager may establish a static assignment at start of a processing of a computation task. Alternatively or additionally it may establish a dynamic assignment at runtime, which means during processing of the computation task.

EX1004, [0013]

(cited by Petition, 11, 39-40)

referred to as RM. The resource manager initializes a first assignment and further on establishes a dynamic assignment of boosters B to computation nodes CN.

EX1004, [0068]

(cited by Petition, 11, 18, 20, 40)

Claims 5 and 9

Claims 5, 9: “daemons”

5. The method according to claim 1, wherein daemons operate in the plurality of computation nodes and the plurality of booster nodes to generate the information.

Petition, 41 (quoting EX1001, 6:11-14)

Prior Art Lippert:

compound of any of them. Especially the resource manager may be formed by any-one of: a microprocessor, a hardware component, a virtualized hardware component or a daemon.

EX1004, [0014]

(cited by Petition, 12, 15, 20, 36, 40, 56; Reply, 21)

Claims 5, 9 – “daemons”

Dr. Long explained:

booster node configured to collect and send this information. POSITAs knew that a daemon is a background process that performs a specific task, and that daemons were commonly implemented at compute nodes to execute tasks at the compute nodes and report status information back to a central resource manager for managing cluster resources. *E.g.*, EX1013, 5-7 (describing a central controller daemon and

EX1003, ¶90 (cited by Petition, 41-42; Reply, 21-22)

plurality of nodes connected through a network to one another”). A POSITA thus would have been motivated to implement background processes (i.e., daemons) operating at each node that communicate the relevant information in a status update for dynamically remapping the sub-tasks among the nodes, 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. *E.g.*, EX1012, 9 (describing a central daemon and node daemons or

EX1003, ¶90 (cited by Petition, 41-42; Reply, 21-22)

Claims 5, 9 – “daemons”

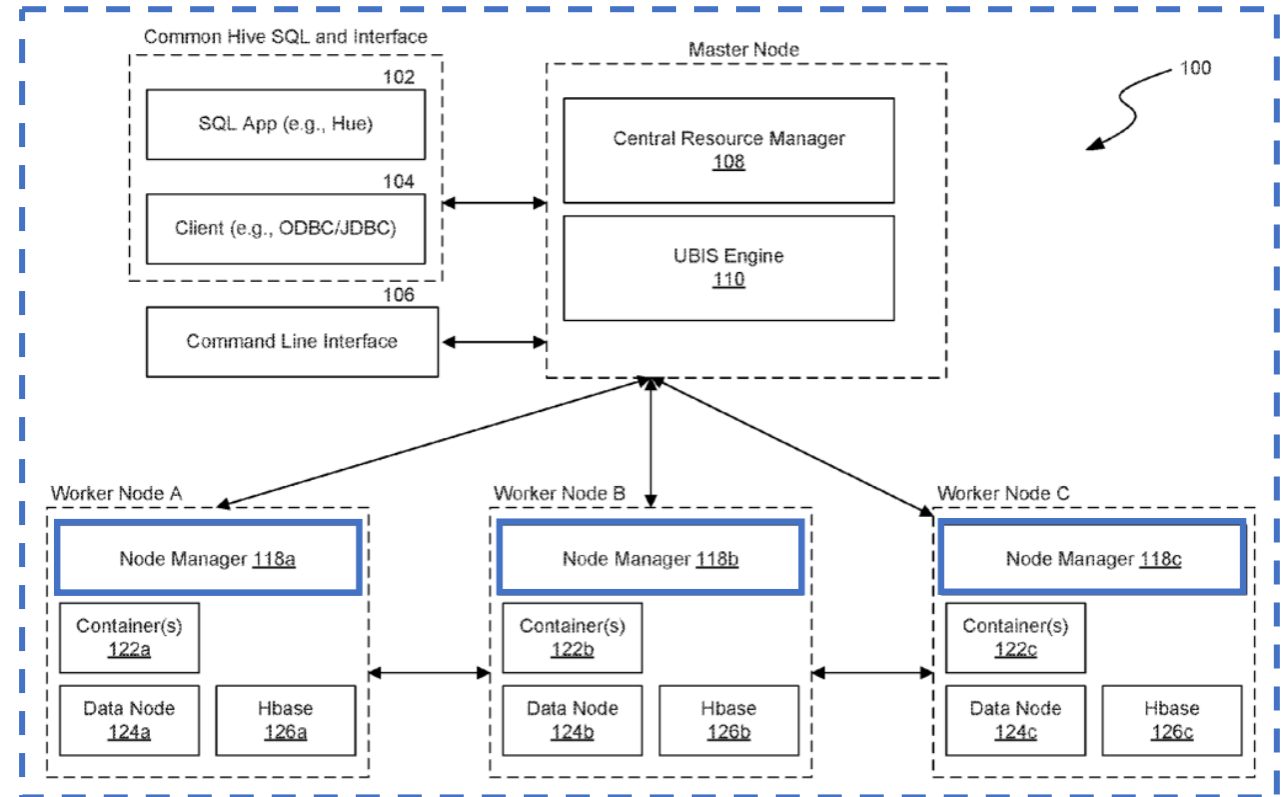
Ground 2: Kambatla discloses a daemon in each worker node:

[0038] According to one embodiment, a resource manager daemon installed at the worker nodes includes a node manager 310b, and one or more application masters 320b. The node manager 310b depicted in FIG. 3B may be the same as or part of node managers 118a-c depicted in FIG. 1.

EX1006, [0038]
(cited by Petition, 59, 68, 70)

[0039] The node manager 310b is a per-node agent installed on each of the worker nodes in the cluster. The node manager 310b includes a node status updater 312b that registers with the resource manager and broadcasts the status of the node including the status of available resources (e.g. containers 330b) at the node. For example, the node status

EX1006, [0039]
(cited by Petition, 69-70)



EX1006, FIG. 1
(cited and annotated by Petition, 68-69)

Appendix

Budenske: dynamic remapping of subtasks in HC system

erogeneous parallel hardware platforms. The contribution of this paper is that, for the computational environment considered, it presents a methodology for real-time on-line input-data dependent remapping of the application subtasks to the processors in the heterogeneous parallel hardware platform using a previously stored off-line statically determined *mapping* (i.e., a matching of application subtasks to processors and a scheduling for the execution order of these subtasks). That is, the operating system will be able to make a heuristically-based decision during the execution of the application whether to perform a remapping based on information generated by the application from its input data. If the decision is to remap, the operating system will be able to select a previously heuristically derived and stored mapping that is appropriate for the given state of the application (e.g., the number of objects it is currently tracking).

EX1005, 387

(cited by Petition, 13, 16-17, 19, 26)

Budenske – “generating, using information relating to the processing”

Budenske teaches generating using (1) pre-determined candidate mappings or (2) on-line heuristic:

“MDDG Table”). *Id.*, 402. Budenske thus teaches selecting a new mapping from a set of candidate mappings that best matches the values of the dynamic parameters (*the information*), and instantiating (*generating*) the further distribution according to the selected mapping for the next computing iteration. *Id.*, 402-03. Budenske also discloses an alternative wherein a fully “on-line heuristic” is used, such that the new mapping is generated in real-time between iterations based on the information. *Id.*, 389-90, 398, 403; EX1003, ¶78.

Petition, 31-32

Budenske: algorithms for the subtasks

For the initial iteration through the set of subtasks, the IOS will use information about the processing environment in its selection of algorithms for the subtasks, and their associated implementations. As part of this, the IOS will decide how to assign resources (e.g., processors) to the subtasks.

EX1005, 391
(cited by Petition, 1, 14, 17-19, 21,
26, 28, 30-32, 34, 38, 40; Reply, 3)

Budenske: an algorithm implementation “for each processor type”

The Algorithm Database will include one or more implementations of each algorithm (e.g., one for each processor type). The Algorithm Database must also contain the expected execution time of each algorithm implementation, typically specified as a function of type and number of processors assigned, interprocessor communication time, and certain static and dynamic Input Data and Application Characteristics. This is an expected time, rather than a definite time, because it may vary depending on the actual values of the input data being processed. This expected execution time information must be provided by

EX1005, 396
(cited by Petition, 28-29, 38;
Reply, 3, 9, 14)

Budenske: “initial iteration through the set of subtasks”

Budenske discloses assigning a plurality of sub-tasks to processing resources of a heterogeneous computing system for processing in a first iteration. For example, Budenske discloses, “[f]or the initial iteration through the set of subtasks” (*in a first computing iteration*), “assign[ing] resources (e.g., processors) to the subtasks.” EX1005, 391; *see also id.*, 396-97 (describing how to assign sub-tasks to processors). This “initial mapping” (*first distribution*) of the subtasks (*the plurality of sub-tasks*) to the processing resources is used for a first computing iteration of the application (*computation task*). *Id.*, 395; EX1003, ¶75.

Petition, 28

Budenske: “the next execution iteration through the subtasks”

EX1005, 395. Budenske thus discloses using values of dynamic parameters (*information relating to the processing of the sub-tasks by at least the portion of the plurality of computation nodes and at least the portion of the plurality of booster nodes*) to generate a “new assignment” (*further distribution*) of sub-tasks to resources “to use for the next execution iteration through the subtasks” (*in a further computing iteration*). *Id.*, 391; EX1003, ¶77.

Petition, 30

Institution Decision Rejected ParTec's Reading of Budenske

Prelim. Resp. 18. We have read the cited portions of Budenske and disagree with Patent Owner's characterization of the cited description. Budenske does not suggest, much less expressly state, that "for its teachings to work," as Patent Owner has characterized, it is assumed that if an implementation of a given subtask uses multiple processors, all processors will be of the same type. The actual quotation is: "For simplicity, it is assumed that if an implementation of a given subtask uses multiple processors, all processors will be of the same type." Ex. 1005, 392. That is far different from Patent Owner's proposed reading, which amounts to disclaiming or nullifying all disclosures of using a combination of different types of processors to perform a subtask. The description does not support Patent Owner's assertion. On this record, in our view, the description is merely providing a

Institution Decision (Paper 10), 21

Lippert-Budenske renders claim 3 obvious under PO's construction

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).

EX1030, ¶51 (cited by Reply, 20-21)

CERTIFICATE OF SERVICE
IN COMPLIANCE WITH 37 C.F.R. § 42.6(E)(4)

The undersigned certifies that on April 10, 2026, a complete copy of the
PETITIONER'S DEMONSTRATIVES were served on counsel for the Patent
Owner via electronic mail as follows:

Michael H. Heim
Christopher L. Limbacher
Heim, Payne & Chorush, LLP
1111 Bagby Street, Suite 2100
Houston, TX 77022
Email: mheim@hpcllp.com
climbacher@hpcllp.com

By: /Andrew M. Mason/
Andrew M. Mason (Reg. No. 64,034)
andrew.mason@klarquist.com
KLARQUIST SPARKMAN, LLP
One World Trade Center, Suite 1600
121 S.W. Salmon Street
Portland, Oregon 97204
Tel: 503-595-5300
Fax: 503-595-5301

Counsel for Petitioner