

Compare Results

2025-04-09 9:28:26 PM

Old File:

**20241220 PAPER 1 IPR2025-00318 Microsoft
Petition for IPR '442_38.pdf**

5 pages (168 KB)
2025-04-09 9:27:32 PM

versus

New File:

Exhibit 1003 - Long Declaration_42.pdf

4 pages (181 KB)
2025-04-09 9:28:08 PM

Total Changes

34

Content

20 Replacements

2 Insertions

7 Deletions

Styling and Annotations

8 Styling

5 Annotations

[Go to First Change \(page 1\)](#)

c) **Element [1.3]**

[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.

77. Lippert-Budenske discloses and teaches this element. After processing the sub-tasks with the “initial mapping,” Budenske teaches “monitor[ing] 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).” 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*). EX1005, 391.

78. Specifically, “[a]fter 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.” EX1005, 391. The dynamic parameters are “computed by the application as it executes.” EX1005, 393. The values for the dynamic parameters are thus obtained after processing the sub-tasks in an iteration: “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.” EX1005, 391. At each iteration, there can be a different mapping of the DDG (parallelism) onto the accelerators. The latest values of dynamic parameters are an example of the claim’s “information relating to the processing of the plurality of sub-tasks.” The “most recent values of such dynamic parameters” are then used to determine whether “to reconfigure the assignment of processing resources to subtasks to reduce execution time of the next iteration,” and “[i]f it is desirable, the HC Kernel will select a new assignment to use for the next execution iteration through the subtasks.” EX1005, 391. In particular, Budenske teaches generating a “vector of D representative values” as an “approximation of the set of current actual dynamic parameters,” and using that vector to identify the optimal mapping of sub-tasks from the possible mappings (tracked in a “Mapped DDG” or “MDDG Table”). EX1005, 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.

EX1005, 402-403. 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. EX1005, 389-90, 398, 403.

79. A POSITA would have found it obvious to implement the heterogeneous computing system of Lippert with the dynamic sub-task mapping method of Budenske such that, between iterations, Lippert-Budenske uses values of dynamic parameters calculated during a preceding iteration to generate a new assignment of sub-tasks to the plurality of computation nodes and the plurality of boosters for processing in the next iteration. The POSITA would have been motivated to implement Lippert-Budenske in this way partly in order “to reduce execution time of the next iteration.” EX1005, 391. The POSITA would have further been motivated to implement Lippert-Budenske in this way because Budenske explains that, “[b]ecause the execution time of application subtasks ... is highly input-data dependent ..., this matching and scheduling of application subtasks to processors *must* be performed dynamically at run time.” EX1005, 390 (emphasis added). The POSITA would have reasonably expected to succeed in implementing Lippert-Budenske partly because Lippert and Budenske each provide ample implementation details for achieving the disclosed functionality, and a POSITA would have found it simple and straightforward to combine the teachings. Lippert-Budenske thus teaches and renders this element and this claimed method obvious.

[9.5] an application manager configured to receive information from daemons operating in at least the portion of the plurality of computation nodes and at least the portion of the plurality of booster nodes to update a distribution of the multiple sub-tasks between the plurality of computation nodes and the plurality of booster nodes in a further computing iteration.

107. Lippert-Budenske discloses and teaches this element. Section X.C.2.

Lippert-Budenske thus discloses and teaches all elements and renders this claimed system obvious.