Internal distribution code:

(A) [-] Publication in OJ
(B) [-] To Chairmen and Members
(C) [-] To Chairmen
(D) [X] No distribution

Datasheet for the decision of 12 January 2018

Case Number: T 0790/14 - 3.5.06

Application Number: 09751296.6

Publication Number: 2297639

IPC: G06F9/50, G06F9/45

Language of the proceedings: EN

Title of invention:
METHOD OF USING PARALLEL PROCESSING CONSTRUCTS

Applicant:
The MathWorks, Inc.

Headword:
Programming language construct/MATHWORKS

Relevant legal provisions:
EPC Art. 56

Keyword:
Inventive step - (no)

Decisions cited:
T 0423/11, T 1539/09

Catchword:

Europäisches Patentamt
European Patent Office
Office européen des brevets

Beschwerdekammern
Boards of Appeal
Chambres de recours

European Patent Office
D-80298 MUNICH
GERMANY
Tel. +49 (0) 89 2399-0
Fax +49 (0) 89 2399-4465

Case Number: T 0790/14 - 3.5.06

D E C I S I O N
of Technical Board of Appeal 3.5.06
of 12 January 2018

Appellant: (Applicant)	The MathWorks, Inc. 3 Apple Hill Drive Natick, MA 01760-2098 (US)
Representative:	Müller-Boré & Partner Patentanwälte PartG mbB Friedenheimer Brücke 21 80639 München (DE)
Decision under appeal:	**Decision of the Examining Division of the European Patent Office posted on 29 November 2013 refusing European patent application No. 09751296.6 pursuant to Article 97(2) EPC.**

Composition of the Board:

Chairman	W. Sekretaruk
Members:	S. Krischer
	A. Teale

Summary of Facts and Submissions

I. The appeal is directed against the decision of the examining division, dated 29 November 2013, to refuse application No. 09751296.6 for added subject-matter (Article 123(2) EPC) and lack of clarity (Article 84 EPC).

II. A notice of appeal was received on 3 February 2014. The appeal fee was paid on the same day. A statement of grounds of appeal was received on 10 March 2014. Claim sets according to a main and an auxiliary request were filed.

III. In a communication dated 27 July 2017, the rapporteur raised an objection of lack of inventive step.

IV. In a letter dated 29 August 2017, the appellant submitted arguments and filed a much redrafted claim set as its sole request.

V. In its summons to oral proceedings, the board mainly gave further reasons why the claims still lacked an inventive step.

VI. In a letter dated 11 December 2017, the appellant submitted further arguments and filed a clarified claim set as its sole request.

VII. Oral proceedings were held on 12 January 2018. At their end, the board announced its decision.

VIII. The appellant requests that the decision under appeal be set aside and that a patent be granted on the basis of the request of 11 December 2017. The other

application documents are the same as in the appealed decision.

IX. Claim 1 reads as follows:

"1. A computing device-implemented method for performing parallel processing, comprising:

receiving (2210), by a client (500), a program (810), wherein the program (810) comprises an SPMD command;

analyzing (2220) and transforming the program (810);

determining (2230) an inner parallel context (830) and an outer parallel context (820) of the program (810) based on the analysis of the program (810),

wherein the inner parallel context (830) and the outer parallel context (820) include boundaries (840) of the program (810),

wherein the boundaries (840) include an spmd statement and an end statement,

wherein the inner parallel context (830) includes a code block within the boundaries (840) and the outer parallel context (820) includes a code block outside the boundaries (840);

executing (2240), by the client (500), the outer parallel context (820) of the program (810) sequentially;

detecting input variables and output variables used within the inner parallel context (830), wherein the input variables include variables used within the inner parallel context before they are assigned values;

allocating (2270) the inner parallel context (830) of the program (810) and the detected input variables to two or more labs (420) for parallel execution, wherein the allocating comprises dividing a large data set into pieces and providing each piece of data to a

different one of the labs (420), such that each lab (420) executes the same program on its piece of data;

wherein each of the labs (420) includes hardware, software or a combination of hardware and software that performs parallel processing,

wherein there is no implicit data transfer to and from the client (500) and the labs (420) that will execute the inner parallel context,

if an input variable to be transferred from the client (500) to the labs (420) is a distributed array, then the variable will be automatically redistributed to the labs (420),

if an input variable to be transferred from the client (500) to the labs (420) is a non-distributed array, then the variable will be replicated on the labs (420)

executing the one or more portions of the inner parallel context on the two or more labs (420);

receiving (2280) one or more results associated with the parallel execution of the inner parallel context from the two or more labs (420),

wherein the names of the output variables are propagated to the outer parallel context but the values associated with the output variables are not copied to the outer parallel context; and

providing (2290) the one or more results to the outer parallel context (820) of the program (810)."

Reasons for the Decision

1. *Summary of the invention*

The application relates to the definition of a specific parallel programming language construct, the so-called

SPMD command (see figure 8 and page 14, line 28, to page 16, line 10; in particular page 15, line 27), including the way it is supposed to be executed on a parallel computer system.

2. *Inventiveness*

2.1 Programming language constructs (i.e. the commands and their syntax) are the main part of the definition of a programming language. The way they are executed also belongs to the definition of a programming language, namely to its operational semantics.

2.2 However, programming language constructs are even more abstract than programs which are as such excluded from patentability. Furthermore, programming is a mental act, and programming language constructs have the intrinsic aim of enabling and easing the work of a programmer which itself lacks technical character (see T423/11, 3.9-3.12 and T1539/09, 4.).

2.3 Therefore, the design and the definition of programming language constructs (including the operational semantics, the data flow, the error handling and side effects) is not considered to contribute to the technical character of claimed subject-matter and thus cannot establish the presence of an inventive step. In the present case, the claimed method merely represents the operational definition of the SPMD command.

2.4 The board finds that the technical effects indicated in the grounds of appeal (page 9, bottom to page 10, first paragraph; page 15, second paragraph) are not convincing:

- "resource allocation is efficient and flexible": the resource allocation is mainly done by the programmer who has to deliver remote references to labs in one of the input variables of the program he/she is writing (see step of detecting input variables in claim 1); see below for a more detailed argumentation; furthermore, there is no technical disclosure of the allocation in the description;
- "it is easy to send ordinary variables into parallel code"/"data is transferred between code executed in parallel and code executed sequentially in an easy and efficient way": this relates to the design and the definition of the SPMD command and therefore does not contribute to the technical character; furthermore, the input variables (i.e. the variables which are used in the SPMD body (= inner parallel context) before they are assigned values; see page 23, second paragraph) anyway have to be transmitted to the labs, since otherwise the labs cannot execute the code in the SPMD body;
- "minimal data transfer via the use of remote reference": this also relates to the operational definition of the SPMD command; furthermore, the word "minimal" needs a point of comparison; however, other parallel programming language constructs are deliberately defined to work differently; they cannot be compared with the present SPMD command in order to show a technical improvement (i.e. a technical effect), since they are defined to do something different; see below for a more detailed argumentation;
- "when an error occurs and parallel execution is interrupted, work can be picked up from where the interruption occurred and need not be resumed from

the beginning of a task": this relates to the design of the error handling in the definition of the SPMD command; furthermore, error handling is no longer set out in the claims.

2.5 The board agrees with the appellant that, in contrast to a mere (non-technical) definition of a programming language construct, a computer-implemented method for *executing* a program *might* produce a technical effect while it executes a program, for example if the new method executes the program faster than a prior art method, or if less data transfer is needed during execution (as is stated in the letter dated 29 August 2017, page 7, second paragraph; for the prior art method see the paragraph bridging pages 7 and 8). However, this effect can only be produced if the new method executes the *same* program as the prior art method.

2.6 In the present case, a second program containing the new programming language construct ("SPMD command") would have to be written by a *human programmer* to perform exactly the same function (i.e. having the same input-output-behaviour) as a first program which does not contain the new construct and can be executed by the prior art executing method.

2.7 Thus, in order to compare the two methods executing the same program, a human programmer is needed to transform one program into another in order to make it executable by the other execution method.

2.8 Therefore, no technical effect is achieved by a new execution method with respect to an existing execution method, if the programming language accepted by each of

the two execution methods is different. This is the case if the new programming language contains at least one new command. A command is new, even if the keywords of the new command are already used (here "SPMD" and "END"; see page 15, line 27), but the command allows additional information to be entered by the programmer (here "IN1" and "OUT1" for the input and output variables; see also page 15, line 27) or is to be executed differently.

2.9 Furthermore, it is this extra work by the programmer (to indicate the input and output variables to be transferred between the client computer and the lab computers) which allows the new execution method to reduce the amount of data to be transferred (see letter of 29 August 2017, page 8, second paragraph).

2.10 Thus, even if the former argument of the impossibility of comparison with an existing execution method were not valid, an inventive step would still not be involved, since the new method merely does what the programmer tells it, namely only transferring the input and output variables which the programmer has indicated. The inventive merit of the alleged reduction in the data transfer would be due to the programmer, and not to the execution method.

2.11 Therefore, the subject-matter of the claims is not inventive.

Order

For these reasons it is decided that:

The appeal is dismissed.

The Registrar: The Chairman:

B. Atienza Vivancos W. Sekretaruk

Decision electronically authenticated