

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

AMERICAN AIRLINES, INC. AND SOUTHWEST AIRLINES CO.,

Petitioners,

v.

INTELLECTUAL VENTURES I LLC,

Patent Owner.

IPR2025-00785

PATENT NO. 7,257,582

DECLARATION OF MICHAEL IAN SHAMOS, PH.D., J.D.

TABLE OF CONTENTS

I. INTRODUCTION AND QUALIFICATIONS	1
II. MATERIALS CONSIDERED	5
III. SUMMARY OF OPINIONS	8
IV. LEGAL PRINCIPLES	8
V. LEVEL OF ORDINARY SKILL IN THE ART	14
VI. CLAIM CONSTRUCTION.....	16
VII.BACKGROUND OF THE PRIOR ART.....	17
A. Yarbrough (1960) (EX1009).....	17
B. Baer (1973) (EX1010).....	19
C. Quinn (1988) (EX1011)	20
D. Hu et al. (1989) (EX1012).....	21
E. Hurson et al. (1989) (EX1013).....	23
F. Deo et al. (1991) (EX1014).....	25
G. Xu et al. (1993) (EX1015).....	25
H. Rüb (1995) (EX1016).....	26
I. Kearsley et al. (1996) (EX1018)	27
J. Blelloch et al. (1997) (EX1017).....	28
K. Koniges et al. (2000) (EX1019)	29
L. Schwiegelshohn et al. (1998) (EX1021)	34
M. Barkes et al. (1998) (EX1020)	35
N. Chow et al. (2001) (EX1003).....	36
O. Reiffin (2001) (EX1004).....	39
P. Kurowski et al. (2011) (EX1005).....	41
Q. Conclusions Regarding the Prior Art	43
VIII. THE CHALLENGED PATENT.....	43
IX. Ground 1: Claims 1-14, Obvious over CHOW IN VIEW OF REIFFIN.....	51
A. Motivation to combine Reiffin (EX1004) with Chow (EX1003).....	51
B. Claim 1	52

1. Preamble.....	53
2. Limitation 1a	54
3. Limitation 1b	56
4. Limitation 1c	58
5. Limitation 1d.....	59
6. Limitation 1e	60
C. Claim 2	62
D. Claim 3	64
E. Claim 4	64
F. Claim 5	66
G. Claim 6	68
H. Claim 7	69
I. Claim 8	70
J. Claim 9	71
K. Claim 10	72
L. Claim 11	73
M. Claim 12	75
N. Claim 13	77
O. Claim 14	78
X. Ground 2: Claims 1-4 were Obvious over CHOW IN VIEW OF REIFFIN, FURTHER IN VIEW OF KUROWSKI.....	80
A. Motivation to Combine	80
B. Claim 1	82
1. Limitation [1pre]	82
A method of effecting on a preexisting input file a computer-executable process comprised of a plurality of subtasks, the method comprising the steps of: 82	
2. Limitation [1a].....	83
(a) automatically determining file allocation and logically subdividing records of said input file into a plurality of partitions;	
3. Limitation [1b]	85

[1b] (b) distributing descriptions of all of said partitions to each of a plurality of subtask processors.....	85
4. Limitation [1c].....	85
[1c] (c) simultaneously executing at least a respective one of the subtasks of the computer-executable process in each of at least some of said processors on a respective one of the partitions with each subtask reading and processing the respective partition so as to process the respective partition and produce respective subtask output and;	86
5. Limitation [1d]	87
[1d] (d) thereafter repeating step (c) in at least some of the subtask processors each with another unprocessed partition on a first-come/first-served basis; and 87	
6. Limitation [1e].....	88
[1e] (e) generating at least one output combining all of the subtask outputs and reflecting the processing of all of said subtasks.....	88
7. Claim 2:	89
2. The method defined in claim 1 wherein the automatic determination of file allocation and logical subdivision of records of said input file into said plurality of partitions in step (a) and the distribution of the description of all of said partitions in step (b) is carried out with at least one further processor in addition to the subtask processors.....	89
8. Claims 3-14	90
XI. CONCLUDING STATEMENT	91
XII. JURAT	91

I, Michael I. Shamos, declare as follows.

I. INTRODUCTION AND QUALIFICATIONS

1. I have been retained by McKool Smith P.C. on behalf of Petitioners American Airlines. Inc. (“American”) and Southwest Airlines Co. (“Southwest”) (collectively “Petitioners”) to provide my opinions concerning the validity of claims 1-14 (the “Challenged Claims”), constituting all claims of U.S. Patent No. 7,257,582 (“’582 Patent,” or the “Patent,” EX1001) in support of Petitioners’ Petition for *Inter Partes* Review of the ’582 Patent (the “Petition”).

2. I have been retained as an independent expert consultant in this proceeding before the United States Patent and Trademark Office. Although I am being compensated at my rate of \$625.00 per hour for the time I spend on this matter, no part of my compensation depends on the outcome of this proceeding, and I have no other interest in this proceeding, any of the parties, or the subject patent.

3. A detailed description of my professional qualifications, including a listing of my specialties/expertise and professional activities and publications, is contained in my curriculum vitae, a copy of which is provided as EX1019. Below is a short summary of my professional qualifications.

4. I received an A.B. (1968) from Princeton University in Physics; a M.A. (1970) from Vassar College in Physics; a M.S. (1972) from American University in Technology of Management, a field that covers quantitative tools used in managing

organizations, such as statistics, operations research and cost-benefit analysis; a M.S. (1973), and M.Phil. (1974) and a Ph.D. (1974) from Yale University in Computer Science; and a J.D. (1981) from Duquesne University.

5. I currently hold the title of Distinguished Career Professor in the School of Computer Science at Carnegie Mellon University in Pittsburgh, Pennsylvania. I am a member of two departments in that School, the Software and Societal Systems Department and the Language Technologies Institute. I was a founder and Co-Director of the Institute for eCommerce at Carnegie Mellon from 1998-2004 and from 2004-2018 was Director of the eBusiness Technology graduate program in the Carnegie Mellon University School of Computer Science. Since 2018, I have been Director of the M.S. in Artificial Intelligence and Innovation degree program at Carnegie Mellon.

6. I have taught graduate courses at Carnegie Mellon in Electronic Commerce, including eCommerce Technology, Electronic Payment Systems, Electronic Voting, Internet of Things, Electronic Payment Systems and eCommerce Law and Regulation, as well as Analysis of Algorithms. Since 2007 I have taught an annual course in Law of Computer Technology. I currently also teach Artificial Intelligence and Future Markets.

7. I am the author and lecturer in a 24-hour video course on Internet protocols and have taught computer networking, wireless communication, and Internet architecture since 1999.

8. From 2001-2021, I was a Visiting Professor at the University of Hong Kong, where I taught an annual course in Electronic Payment Systems. This is one of only a handful of graduate courses taught on this subject in the world.

9. I was the Director of Carnegie Mellon's graduate degree program in eBusiness Technology from 1999-2018 and am now a faculty member in the Privacy Engineering degree program at Carnegie Mellon. My course on Law of Computer Technology is required for all students in that program, the M.S. in Artificial Intelligence and Innovation program and in the Societal Computing Ph.D. program.

10. From 1979-1987, I was the founder and president of two computer software development companies in Pittsburgh, Pennsylvania, Unilogic, Ltd. and Lexeme Corporation.

11. I am an attorney admitted to practice in Pennsylvania and have been admitted to the Bar of the U.S. Patent and Trademark Office since 1981. I have been asked to render opinions in this Declaration as a technical expert. I have not been asked to offer any opinions on patent law in this proceeding.

12. I am a named co-inventor on the following six issued patents relating to electronic commerce: U.S. Patent Nos. 7,330,839, 7,421,278, 7,747,465, 8,195,197, 8,280,773, and 9,456,299.

13. I have previously served as an expert in over 400 cases concerning computer technology.

14. I am aware of contemporaneous litigation involving the parties in two cases, styled *Intellectual Ventures I LLC et al. v. American Airlines, Inc.*, 4:24-cv-980-ALM (E.D. Texas) and *Intellectual Ventures I LLC et al. v. Southwest Airlines Co.*, No. 7:24-cv-277 (W.D. Tex. Nov. 2, 2024) (the “Litigations”).

15. This IPR petition involves the ’582 Patent, which was filed on February 27, 2003 and entitled “Load Balancing with Shared Data.” The Patent claims priority to U.S. Provisional Patent Application 60/363,853 (the “’853 Provisional”), filed March 13, 2002. The term of the Patent was extended by 912 days, so it will expire on August 27, 2025.

16. I have been asked to assume, for purposes of this IPR proceeding only, that the claims of the Patent are all entitled to a Priority Date of March 13, 2002, although it is apparent from an inspection of the one-page ’853 Provisional application that it does not support any Challenged Claim.

17. I have no financial interest in Patent Owner or the ’582 Patent. It is conceivable that I may own mutual funds whose portfolios include stock in one or

more Petitioners. If this is the case, the value of such holding would not constitute a material part of my net worth.

18. The statements made and opinions provided in this Declaration are based on my own personal knowledge and, if called as a witness, I could and would testify in a manner consistent with this Declaration.

19. In this Declaration, all emphasis has been added unless otherwise noted.

II. MATERIALS CONSIDERED

20. In forming my opinions, I have reviewed and am familiar with the '582 patent and its prosecution history, the materials cited herein, and the following exhibits to the Petition.

Exhibit No.	Description
1001	U.S. Patent No. 7,257,582 to Michael Rothschild
1002	Prosecution history of the '582 Patent
1003	U.S. Patent No. 6,304,866 to Chow et al. (Chow)
1004	U.S. Patent No. 6,330,583 to Reiffin (Reiffin)
1005	U.S. Patent Application No. 2002/0019844 of Kurowski et al. (Kurowski)
1006	Complaint in <i>Intellectual Ventures I LLC et al. v. American Airlines, Inc.</i> , No. 4:24-cv-980-ALM (E.D. Tex. Nov. 2, 2024)
1007	Claim Chart for '582 Patent, <i>Intellectual Ventures I LLC et al. v. American Airline, Inc.</i> , No. 4:24-cv-980-ALM (E.D. Tex. Nov. 2, 2024) (Dkt. 1-14) (Hadoop)

Exhibit No.	Description
1008	Claim Chart for '582 Patent, <i>Intellectual Ventures I LLC et al. v. American Airline, Inc.</i> , No. 4:24-cv-980-ALM (E.D. Tex. Nov. 2, 2024) (Dkt. 1-15) (Spark)
1009	"Some Thoughts on Parallel Processing," Lynn D. Yarbough, Communications of the ACM, Vol. 3, Issue 10, page 539, October 1, 1960 (available at: https://dl.acm.org/doi/10.1145/367415.367426).
1010	"A Survey of Some Theoretical Aspects of Multiprocessing," J. L. Baer, ACM Computing Surveys (CSUR), Volume 5, Issue 1, Pages 31 – 80, March 1, 1973 (available at: https://dl.acm.org/doi/10.1145/356612.356615)
1011	"Parallel Sorting Algorithms for Tightly Coupled Multiprocessors," Michael J. Quinn, Parallel Computing 6 (1988) 349-357.
1012	"Removing Skew Effect in Join Operation on Parallel Processors," Ron-Chung Hu, Richard R. Muntz, Computer Science Technical Report, University of California, Los Angeles, June 1989.
1013	"Parallel Architectures for Database Systems," A.R. Hurson, L.L. Miller, S.H. Pakzad, M.H. Eich, B. Shirazi, Advances in Computers, Vol. 28, pages 107-151, 1989.
1014	"Parallel Algorithms for Merging and Sorting," Narsingh Deo, Dilip SarKar, Information Sciences, Vol. 56, pages 151-161, 1991.
1015	"Parallel Algorithms for least Median of Squares Regression," Chong-Wei Xu and Wei-Kei Shiue, Computational Statistics & Data Analysis, Vol. 16, pages 349-362, 1993.
1016	"On the Average Running Time of Odd-Even Merge Sort," Christine Rub, Max Plank-Institute Fur Informatik, Research Report, April 1995.
1017	"Parallel Algorithms," Guy E. Blelloch and Bruce M. Maggs, a chapter in "The Computer Science and Engineering Handbook." CRC Press, 1997, ISBN: 0-8493-2909-4. Available at https://dl.acm.org/doi/pdf/10.5555/1882723.1882748 .

Exhibit No.	Description
1018	“An Approach to Parallelizing Isotonic Regression,” Anthony J. Kearsley, Richard A. Tapia, Michael W. Trosset, Applied Mathematics and Parallel Computing, H. Fisher et. al. (eds.), Physica-Verlag Heidelberg (1996), pages 141-147.
1019	“Part 1: The Parallel Computing Environment, Alice E. Koniges, Morris A. Jette, and David C. Eder, in “Industrial Strength Parallel Computing,” Morgan Kaufmann, (2000). ISBN 9781558605404. Available at: http://wayback.cecm.sfu.ca/PSG/book/intro.html and at https://books.google.com/books?id=mWalawBciCQC&pg=PA1&source=gbs_toc_r&cad=2#v=onepage&q&f=false .
1020	GPFS: A Parallel File System, Jason Barkes, Marcelo R. Barrios, Francis Cougard, Paul G. Crumley, Didac Marin, Hari Reddy, Theerapong Thitayanun, IBM International Technical Support Organization, April 1998.
1021	“Analysis of First-Come-First-Serve Parallel Job Scheduling,” Uwe Schwiegelshohn and Ramin Yahyapour. Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (1998). Available at http://dx.doi.org/10.1145/314613.315031 .
1022	This Declaration
1023	CV of Dr. Michael Shamos
1024	Claim Chart for '582 Patent, <i>Intellectual Ventures I LLC et al. v. Southwest Airlines Co.</i> , No. 7:24-cv-277 (W.D. Tex. Nov. 2, 2024) (Dkt. 1-14) (Hadoop)
1025	Claim Chart for '582 Patent, <i>Intellectual Ventures I LLC et al. v. Southwest Airlines Co.</i> , No. 7:24-cv-277 (W.D. Tex. Nov. 2, 2024) (Dkt. 1-15) (Spark)

21. I have also relied on my education, skill, training, and experience in the relevant fields of technology. I have further considered the viewpoint of a person of

ordinary skill in the art (“POSITA”) in the subject matter of the ’582 Patent as of March 13, 2002.

22. I reserve the right to supplement my opinions as expressed in this Declaration to address any new information obtained in the course of this proceeding, or based on any new positions taken by Patent Owner.

III. SUMMARY OF OPINIONS

23. It is my understanding that the following table summarizes the grounds of challenge (“Grounds”) raised in the Petition:

Grd.	Sec.	References	Challenged Claims
1	103	Chow (Ex.1003) in view of Reiffin (Ex.1004)	1-14
2	103	Chow (Ex.1003) in view of Reiffin (Ex.1004) and further in view of Kurowski (Ex.1005)	1-14

24. After a review of the ’582 Patent and the prior art asserted by Petitioners, it is my opinion that the Challenged Claims are invalid under both Grounds. My opinions, and the bases therefor, are detailed throughout this Declaration.

IV. LEGAL PRINCIPLES

25. Counsel for Patent Owner has informed me of the legal principles that apply in this proceeding.

26. I understand that a claim is unpatentable for obviousness under 35 U.S.C. § 103 “if the differences between the subject matter sought to be patented

and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains.” Obviousness may be based upon a combination of references. Obviousness may also be based on a single reference combined with the knowledge of a person of ordinary skill in the art (“POSITA”) at the time the filing of the application that led to patent.

27. A prior art reference can disclose a limitation that is not expressly disclosed in the reference if it is “inherently present” in the reference. I further understand that to be “inherently present,” the missing disclosure must necessarily and inevitably be present in the element or process described in the reference. It is my understanding that a limitation is not disclosed by inherency if the missing limitation is only probably present or if there is merely a possibility that it is present.

28. For a claim to be obvious in view of a combination of known elements, there must be some apparent reason to combine the known elements in the fashion of the claim at issue by considering the teachings of prior art references, the effects of demands known to people working in the field or present in the marketplace, and the background knowledge possessed by a POSITA. Identifying a reason the elements should be combined is important because inventions in many instances rely upon building blocks long since uncovered, and claimed discoveries almost of necessity will be combinations of what, in some sense, was already known. The prior

art references themselves may provide a suggestion, motivation, or reason to combine, but the nexus linking two or more prior art references is sometimes simple common sense.

29. An obviousness inquiry requires consideration of the following factors: (1) the scope and content of the prior art; (2) the differences between the claims and the prior art; (3) the level of ordinary skill in the pertinent art; and (4) any objective indicia of non-obviousness, such as commercial success, long-felt but unresolved need, failure of others, industry recognition, copying, and unexpected results.

30. Such secondary considerations include: (a) commercial success of a product due to the merits of the claimed invention; (b) a long-felt, but unmet need for the invention; (c) failure of others to find the solution provided by the claimed invention; (d) deliberate copying of the invention by others; (e) unexpected results achieved by the invention; (f) praise of the invention by others skilled in the art; (g) the taking of licenses under the patent by others and (h) the patentee proceeded contrary to the accepted wisdom of the prior art. Secondary considerations are relevant where there is a connection, or nexus, between the evidence and the claimed invention. I am not aware of any such objective evidence of non-obviousness of the Challenged Claims at this time.

31. In addition, I understand that the obviousness inquiry should not be performed in hindsight, but must be performed using the perspective of a person of ordinary skill in the relevant art as of the effective filing date of the patent claims.

32. Obviousness may also be shown by demonstrating that it would have been obvious to modify what is taught in a single piece of prior art to create the patented invention. Obviousness may be shown by establishing that it would have been obvious to combine the teachings of more than one item of prior art. In determining whether a piece of prior art could have been combined with other prior art or with other information within the knowledge of one of ordinary skill in the art, the following are examples of approaches and rationales that may be considered:

(A) Combining prior art elements according to known methods to yield predictable results; (B) Simple substitution of one known element for another to obtain predictable results; (C) Use of a known technique to improve similar devices (methods, or products) in the same way; (D) Applying a known technique to a known device (method, or product) ready for improvement to yield predictable results; (E) Applying a technique or approach that would have been “obvious to try” (i.e., choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success); (F) Known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces if the variations would have been predictable to

one of ordinary skill in the art; or (G) Some teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings to arrive at the claimed invention. I also understand this suggestion or motivation may come from such sources as explicit statements in the prior art, or from the knowledge or common sense of one of ordinary skill in the art.

33. I understand that an invention that might be considered an obvious variation or modification of the prior art may be considered non-obvious if one or more prior art references discourages or lead away from the line of inquiry disclosed in the reference(s). A reference does not “teach away” from an invention simply because the reference suggests that another embodiment of the invention is better or preferred. My understanding is that for a reference to teach away the reference must clearly indicate that the combination should not be attempted (e.g., because it would not work or explicit statement saying the combination should not be made).

34. I further understand that obviousness must be determined in light of all of the facts, and that there is no rule that if a reference teaches away a finding of nonobviousness is mandated. I understand this to be the rule because, for example, a given course of action often has simultaneous advantages and disadvantages, and thus teaching away does not necessarily obviate motivation to combine. Thus, I understand that where the prior art contains apparently conflicting teachings (for

example, where some references teach the combination and others teach away from it) each reference must be considered for its power to suggest solutions to a POSITA considering the degree to which one reference might accurately discredit another. I am informed that the degree of teaching away depends upon the particular facts of each circumstance and that rigid rules that prevent the application of common sense are not mandated.

35. I further understand that obviousness also requires that a POSITA would have perceived that the proposed combination would have had a reasonable expectation of success. I am informed that reasonable expectation of success does not mean certainty of success. If a technique has been used to improve one device, and a person of ordinary skill at the time of invention would have recognized that it would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond their skill.

36. Practical and common-sense considerations should guide a proper obviousness analysis, because familiar items may have obvious uses beyond their primary purposes. A POSITA looking to overcome a problem will often be able to fit together the teachings of multiple prior art references. An obviousness analysis therefore takes into account the inferences and creative steps that a person of ordinary skill would have employed at the time of invention.

37. A proper obviousness analysis focuses on what was known or obvious to a POSITA at the time of invention, not to the patentee. Accordingly, any need or problem known in the field of endeavor at the time of invention and addressed by the patent can provide a reason for combining the elements in the manner claimed.

38. Prior art teachings are properly combined if one of ordinary skill, having the understanding and knowledge reflected in the prior art and motivated by the general problem facing the inventor, would have been led to make the combination of elements recited in the claims. Under this analysis, the prior art references themselves, or any need or problem known in the field of endeavor at the time of the invention, can provide a reason for combining the elements of multiple prior art references in the claimed manner.

39. I understand that in an *inter partes* review, the petitioner shall have the burden of proving a proposition of unpatentability, including a proposition of obviousness, by a preponderance of the evidence.

V. LEVEL OF ORDINARY SKILL IN THE ART

40. I understand that the Board will be asked to characterize a person having ordinary skill in the art (“POSITA”) at the time the purported invention was made. To make that determination, it is my understanding that the Board considers the following factors: (i) the educational level of active workers in the field, (ii) the type of problems encountered in the art, (iii) the prior art solutions to those problems,

the rapidity with which innovations are made, and (v) the sophistication of the technology in the art. It is my understanding that the importance of each factor varies from case to case.

41. The '582 Patent defines its field of invention as follows:

a way to split the performance of a given task among a plurality of processing units which can all access, directly or indirectly, the input data and the devices on which the output data is to be stored.

EX1001, 1:14-17.

42. The specification of the Patent ("Specification") discloses dividing a computational task into partitions, distributing work on the partitions to possibly heterogeneous processors, having the processors perform their work simultaneously, then merging the outputs of the processors to yield a result of the overall computation. The processors may avail themselves of storage-area networks or network-attached storage. EX1001, 2:28-38. The Specification states:

The principal object of the present invention is to enable the decomposition of a certain type of linear processes that currently use a single computer, into equivalent parallel processes that can efficiently use any number of potentially heterogeneous computers, taking the available capacity of each of these computers into account while optimizing execution.

EX1001, 1:48-54

That is, the field of the invention is parallel computing.

43. The claims are drawn to the disclosed process in largely functional terms; that is, they do not specify the details of how any actual computation is to be carried out.

44. In my opinion, in order to understand the Specification and make and use the claimed invention without undue experimentation as of March 13, 2002, one of ordinary skill in the art of the '582 Patent would have had a bachelor's degree in computer science, computer engineering, electrical engineering, or related field, and at least two years of experience in computer networking and parallel computing, or a person with a master's degree in one of the foregoing and at least one year of experience in the aforementioned fields. Additional education could substitute for professional experience, and vice-versa.

45. I had at least the background of such a POSITA at least as early as March 13, 2002.

VI. CLAIM CONSTRUCTION

46. I understand that Patent Owner does not believe any claim construction is necessary in this proceeding, and I concur.

47. I have accorded all of the claim terms their plain and ordinary meaning to a POSITA.

VII. BACKGROUND OF THE PRIOR ART

48. In this section, I discuss the state of the art prior to the Patent as of March 13, 2002. As a POSITA is presumed to have been familiar with all relevant prior art, a POSITA would have been familiar with the material in this section.

49. I find it incredible that the Patent issued. It describes and claims conventional parallel processing of a type that was well-known at least three decades before the Priority Date.

50. The application enjoyed a brief prosecution. The inventor did not disclose a single item of prior art to the Patent Office. The only rejection, on December 8, 2006, resulted in successful amendments to overcome the threadbare prior art of record.

A. Yarbrough (1960) (EX1009)

51. While the early seeds of parallel computing, that is, the use of a plurality of computers or processors to work on a problem simultaneous (in parallel) were sown in the 1950s, the basic structure of the now-claimed invention was laid out in a short paper by Yarbrough in 1960 (EX1009, “Yarbrough”) that was published in the Communications of the Association for Computing Machinery (“CACM”) 3(10): 539. Yarbrough (EX1009) notes that

In the past two years or so I have seen a number of papers and heard a number of talks describing the characteristics of (and the wonders inherent in) certain computers like Gamma 60,

LARC, H-800, STRETCH, and others. Of course, all these machines share a capacity for parallel asynchronous multiple processing. Now this is a truly marvelous property ...

52. Yarbrough (EX1009) continues:

Therefore I offer the following hardware scheme, for what, it is worth. It does not solve all the problems, but I believe it does move the solutions to the biggest ones within the reach of the programmer. Assume a system of n processors, operating in parallel and sharing a common fast-access memory with binary address logic. Allow one processor, called the Monitor (M), access to all of storage. Then allocate storage to the remaining $n-1$ processors (A, B, C, ...) according to the following scheme: Assign the leftmost $n-1$ bits of the address as allocation selectors so that if the i th bit is a 1, the i th processor has access to that cell, and if that bit is a 0, that cell is not available to the i th processor.

53. The Monitor processor is separate from the $n-1$ other processors.

54. Yarbrough (EX1009) then lays out how the parallel computation is to be done:

Consequences of this scheme would be:

1. One module of storage would be unassailable by processors A, B, C. Similarly, one module accessible to A could not be ruined by programs on processors B and C; and so forth.

2. Each of A, B, C would have access m half of storage.

3. Any of the processors could communicate with any others through the appropriate module.

4. Any priority method (that I have been able dream up) can be handled easily within this scheme. (But of course I don't have a really objective point of view.

5. The scheme would, I believe, allow parallel execution of programs which use dynamic storage allocation (e.g., list logic) without resorting to the use of the Monitor as referee, and with a minimum of headaches.

A variant of this scheme would be to allow free use of memory for readout, and use the memory allocation scheme to restrict writing only. This would keep the safety features intact and allow freer communication between processors, but would complicate to some degree the task of storage assignment programs such as Compilers and Assemblers.

55. Yarbrough's Monitor processor corresponds to the "at least one further processor in addition to the subtask processors" of claim 2 and the "at least one processor" of claim 14. EX1009.

B. Baer (1973) (EX1010)

56. By 1973, so much work had gone on in parallel processing that Baer (EX1010, "Baer") wrote a 50-page overview of the field, citing 182 previous papers.

Baer describes both homogeneous (identical) and non-homogeneous parallel processors. EX1010, p. 32. Baer discloses the use of parallel processing for sorting (*id.*, p. 64). A POSITA would recognize this disclosure of Baer as corresponding to the “sort process” of claim 8 of the Patent.

C. Quinn (1988) (EX1011)

57. In 1988, Quinn devoted an entire article to parallel sorting methods, entitled “Parallel sorting algorithms for tightly coupled multiprocessors” (EX1011, “Quinn”). It discloses:

We present three parallel sorting algorithms suitable for implementation on tightly coupled multiprocessors and compare their performance on the Denelcor HEP. Two of the algorithms implemented -- parallel Shellsort and quickmerge -- are new. Shellsort is amenable to parallelization; however, since Shellsort has higher complexity than quicksort, parallel Shellsort is inferior to parallel quicksort. A second new parallel algorithm, called quickmerge, is based upon both quicksort and mergesort. Our implementation of quickmerge achieves significantly higher speedup than our implementation of parallel quicksort.

EX1011, Abstract

58. A POSITA would recognize that Quinn discloses the whole of claim 8 of the Patent and that Quinn’s quickmerge discloses the whole of claim 3 of the Patent.

D. Hu et al. (1989) (EX1012)

59. In 1989, Hu et al. published a paper on parallel database queries entitled, “Removing Skew Effect in Join Operation on Parallel Processors” (“Hu,” EX1012, “Hu”). “Join” is a database query primitive that is particularly time-consuming to execute, and thus benefits from parallel processing.

60. Hu further discloses:

With the advent of low cost computer communication and microprocessor technologies, database systems using multiple microprocessors in parallel have become technically and economically feasible. In recent years, there have been several successful prototypes and commercial systems making use of parallel processing in database operations. Typical examples are MCC's Bubba system [Bora88], University of Wisconsin's Gamma machine [Dewi88], and Teradata's DBC/1012 [Tera88]. All these systems horizontally fragment a data relation across the processors based on the range of primary key value [Dewi88], hashing mechanisms [Dewi88] [Tera88], or access frequencies [Cope88]. Based on this mechanism, database operations, like select, project, and join, can be performed in parallel and their response times can be significantly reduced. The performance improvement on the parallel processor architecture is (ideally) proportional to the number of parallel processors involved in the operation when the data are evenly distributed across the processors.

EX1012, p. 2

61. Hu discloses the conventional step of combining the results of separate computations in a final merge step:

*Usually a hashing scheme is applied to the join attribute for attaining a more uniform distribution of processor workload. Because the same hashing function is used with both join relations, the tuples after redistribution only have to be joined with those tuples on the same processor; hence making each processor perform a join on its disjoint subset in parallel. **The results from each processor then are merged in the final processing step.***

EX1012, p. 5

62. Hu is specific that the processing includes database queries:

*If we want to list the names of all the courses to be offered, **we have to perform a query** by joining Courserelation and Student relation on the CourseId attribute. This query will have single skew since one relation, Student relation, has a skewed distribution in the join attribute.*

EX1012, p. 4

*If the join operation produces many tuples or the join operation is just an intermediate step, i.e. the output of join operation will be used as input of **other operations in the query processing**,*

then we want each processor to generate the same number of tuples as a result of the join operation.

EX1012, p. 23

63. A POSITA would recognize this disclosure of Hu as corresponding to the “database query” of claim 11 of the Patent.

64. Hu also cites 18 previous papers on parallel database query processing.

E. Hurson et al. (1989) (EX1013)

65. In 1989, Hurson et al. published a paper entitled “Parallel Architectures for Database Systems” (EX1013, “Hurson”). It categorizes parallel processors into several groups:

Since the late 1960s, different researchers have attempted to classify various proposed or designed computer architectures (Feng, 1976; Flynn, 1966; Handler, 1982). Flynn (1966) has classified the concurrent space according to the multiplicity of instruction and data streams. According to Flynn’s classification the computer architectures fall into four groups: SISD (Single Instruction stream-Single Data stream), SIMD (Single Instruction stream-Multiple Data stream), MISD (Multiple Instruction stream-Single Data stream) and MIMD (Multiple Instruction stream-Multiple Data stream). ... Händler (1982) has defined the concurrent space as a three-dimensional space $t = (k, d, w)$ in which k is the number of control units interpreting a program. d is the number of arithmetic and logic units (ALUs)

controlled by a control unit. w is the word length or number of bits handled in one of the ALUs.

EX1013, p. 110

66. A POSITA would recognize Händler's (1, n, w) machine as a single multiprocessor having n processors (able to handle words of length w) as the "single multiprocessor" of claim 13 of the Patent.

67. Hurson goes on to describe a classification of parallel computers known as "Rosenthal's Classification" from 1997:

This classification categorizes the database machines according to their functions in a computing system into three groups: Smart peripheral systems, large backend, and distributed network data node (Rosenthal, 1977). A smart peripheral system at the low end is an intelligent controller for conventional rotating storage, which mainly moves some of the access method functions from the mainframe to the controller. At the high end it represents a controller capable of handling database operations at the secondary storage level. Such an architecture can be organized to perform database operations on a unit of data (i.e., data on the fly) or on several units of data (i.e., logic per track). This concept is a solution for the data communication problem, which eliminates one aspect of the 90-10% rule.

EX1013, p. 112

68. A POSITA would recognize that this description describes a parallel architecture in which one processor is a mainframe computer and the other processors are processors of at least one other computer, as claimed in claim 12 of the Patent.

F. Deo et al. (1991) (EX1014)

69. In 1991, Deo et al. published an article in Information Sciences entitled “Parallel Algorithms for Merging and Sorting” (EX1014, “Deo”). It discloses:

Given two sorted arrays $A = \langle a_1, a_2, \dots, a_m \rangle$ and $B = \langle b_1, b_2, \dots, b_n \rangle$, and a parallel machine with p processors, our goal is to merge A and B into a single array C of size $N = m + n$, efficiently. In order to perform merging in parallel, we first devise three different parallel algorithms for partitioning the given arrays A and B into p pairs of subarrays of approximately equal size. Each partitioning algorithm gives rise to a distinct merging algorithm. ... Finally, we discuss how these merging algorithms can be used for sorting in parallel, efficiently.

EX1014, Abstract

70. A POSITA would recognize that Deo’s parallel merging discloses the whole of claim 3 of the Patent and that Deo’s use of parallel merging to sort discloses the whole of claim 8 of the Patent.

G. Xu et al. (1993) (EX1015)

71. In 1993, Xu et al. published a paper entitled “Parallel algorithms for least median of squares regression” (EX* Xu, “Xu”) in the journal *Computational Statistics & Data Analysis*. It describes various parallel computation methods for statistical problems:

Recently much effort has been devoted to devising robust regression procedures to which the least median of squares (LMS) regression proposed by Rousseeuw (Journal of the American Statistical Association 79, 1984) belongs. LMS's breakdown point achieves the highest possible value of 50%. It may not be possible to write down closed form expressions for the estimates of LMS and the computations involved are not trivial. Efforts have been made to look for better performance algorithms. ... This paper reports the use of parallel computing techniques to speedup the computation of LMS. The Rousseeuw's algorithm and the T-sweep algorithm, an algorithm based on Tukey's idea, have been successfully parallelized, but parallel sweep-line algorithm and its refinement may not work at all in the sense that its performance is intolerantly low due to severe load imbalancing.

EX1015, Abstract

72. Because LMS is a statistical analysis problem, a POSITA would recognize that EX1015 discloses the use of parallel computation for “a statistical analysis process,” as claimed in claim 9 of the Patent.

H. Rüb (1995) (EX1016)

73. In 1995, Rüb published a technical report entitled “On the Average Running Time of Odd-Even Merge Sort” (EX1016, “Rüb”). It discloses:

*This paper is concerned with the average running time of Batcher's odd-even merge sort **when implemented on a collection of processors**. We consider the case where n , the size of the input, is an arbitrary multiple of the number p of processors used.*

74. EX1016, Abstract

*We use the following argument. Consider the odd-even merging network for two sorted lists of $p/2$ elements each. It consists of $\log p$ steps where elements are compared and rearranged. **We can use this procedure to merge two sorted lists of length m each (m a multiple of $p/2$) using p processors** by substituting all comparisons (and exchanges) between two elements by splits of two sorted lists at the median of their union (for details see below). If this is done in a straightforward way (that is, two processors that communicate exchange all their elements), the running time will be $O((m/p)\log p)$.*

EX1016, p. 3

75. A POSITA would recognize the merge step of Ex.1016 as corresponding to the “step of merging said subtask outputs” of claim 3 of the Patent. Of course, EX*Rub also discloses the “sort process” of claim 8 of the Patent.

I. Kearsley et al. (1996) (EX1018)

76. In 1996, Kearsley et al. published a book chapter entitled “An approach to parallelizing isotonic regression” (EX1018, “Kearsley”). The chapter discloses:

Isotonic regression is the problem of fitting data to order constraints. We demonstrate that the isotonic regression of a finite set of numbers Y can be obtained by decomposing Y into subsets, performing parallel isotonic regressions on each subset, then performing a trivial isotonic regression on the resulting combined set. Numerical experiments confirm the efficacy of this approach.

EX1018, Abstract

77. Isotonic regression is a well-known statistical problem. A POSITA would recognize that EX1018 discloses the whole of claim 9 of the Patent.

J. Blelloch et al. (1997) (EX1017)

78. In 1997, my colleagues Guy Blelloch and Bruce Maggs at Carnegie Mellon University published a book chapter entitled “Parallel Algorithms” (EX1017, “Blelloch”). It appeared in “The Computer Science and Engineering Handbook. CRC Press, 1997, ISBN: 0-8493-2909-4.” It discusses various ways parallel processors can access memory:

Multiprocessor models can be classified into three basic types: local memory machine models, modular memory machine models, and parallel random-access machine (PRAM) models. Figure 1 illustrates the structure of these machine models. A local memory machine model consists of a set of n processors

each with its own local memory. These processors are attached to a common communication network. A modular memory machine model consists of m memory modules and n processors all attached to a common network. An n -processor PRAM model consists of a set of n processors all connected to a common shared memory.

EX1017, p. 25-3

79. A POSITA would recognize the m networked memory modules of the modular memory model as corresponding to the “storage area network” of claim 6 of the Patent.

80. A POSITA would recognize the n -processor PRAM model as corresponding to the “network attached storage” of claim 7 of the Patent.

81. EX1017 also discloses load balancing of partitions:

Load balancing: A third use of randomness is load balancing. One way to quickly partition a large number of data items into a collection of approximately evenly sized subsets is to randomly assign each element to a subset. This technique works best when the average size of a subset is at least logarithmic in the size of the original set.

EX1017, p. 25-14

K. Koniges et al. (2000) (EX1019)

82. In 2000, Koniges et al. published a book entitled “Industrial Strength Parallel Computing.” The first chapter of that book is in EX1019 (“Koniges”). It describes a wide variety of processor and memory organization used in parallel computing. It first explains:

Parallel computing became a significant subfield of computer science by the late 1960's, but only by the late 1980's had hardware and software technology improved sufficiently for parallel computing to affect the day-to-day lives of applications programmers. As such, it spurred the development of a whole new way of thinking about programming for applications. Rather than deal with the old textbook picture of a flow-charted program going from one step to the next in a sequential fashion, the availability of parallel computers has forced us to rethink our algorithms at every step to determine which tasks can be performed independently and simultaneously with others. In the early days of parallel computing, this transition to parallel algorithms and the associated tools and programming languages remained primary in the research and university setting. As introduction of massively parallel computers based on workstation or commodity chips surfaced, however, these machines that are capable of gigaflop performance and terabyte data storage enticed industry into the growing field of parallel computing.

EX1019, Sec. 1.1

83. It goes on to describe multiple parallel architectures:

Historically, the most common classification scheme for parallel computers (due to Flynn) is based on a 4-letter mnemonic where the first two letters pertain to instructions that act on the data and the second two letters refer to the data stream type [Fin72].

These are:

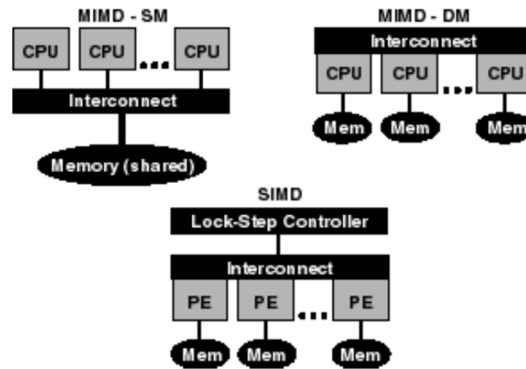
- SISD: Single Instruction stream --- Single Data stream. This is the traditional Von Neuman model of computation where one set of instructions is executed sequentially on a stream of data.*
- SIMD: Single Instruction stream --- Multiple Data stream. These include some of the very early parallel computers such as the Connection Machine, which was manufactured by a company called Thinking Machines. In the SIMD model, multiple individual processors or a single vector processor execute the same instruction set, on different data sets.*
- MISD: Multiple Instruction stream --- Single Data stream. This is not currently a commercially viable system for computing.*
- MIMD: Multiple Instruction streams --- Multiple Data stream. This is the model of parallel computing that has dominated for most industrial applications. Here, multiple processors can be executing different instructions on their own data, and communicating as needed.*
 - For MIMD machines, we have two subclasses, either Shared Memory (SM) or Distributed Memory (DM).*

EX1019, Sec. 2.1

84. The reference then displays examples of various parallel architectures:

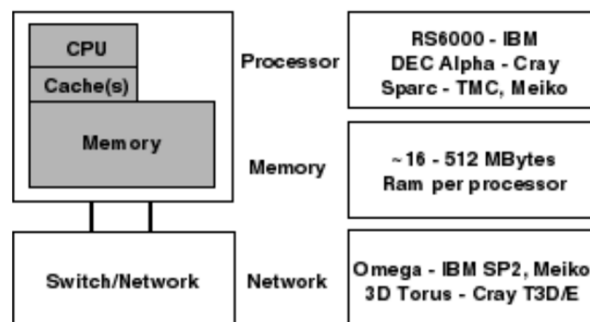
A diagram showing the basic difference between the MIMD-SM and MIMD-DM is given in Figure 1.

Fig. 1



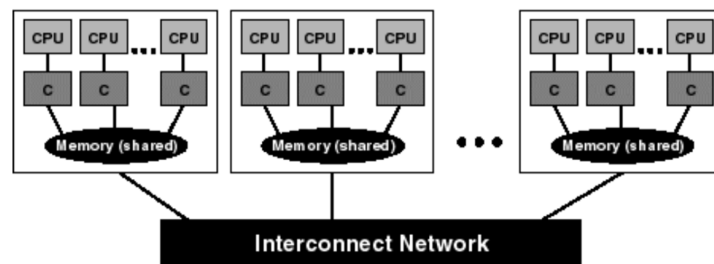
For the simplest description of an MPP computer, one can partition it into processors, memory with various access levels including cache(s), interconnect network or switch, and support hardware. (See Figure 2.) Some details of these various components of an MPP computer are discussed in the following subsections.

Fig. 2



One of the latest developments in high performance computing is the coupling of SMP machines into clusters using very high speed interconnects. These clusters can have total number of processors that are comparable to large MPP machines. A typical cluster configuration is shown in Figure 3. They present the application developer the benefit of relatively large shared memory within a given SMP and total memory comparable to MPP machines. The clustering concept is not restricted to joining SMPs, many architectures are suitable for nodes of a cluster, and the nodes need not necessarily be all of the same type. This is often referred to as a "convergence of architectures" in the development of high-end computing.

Fig. 3



85. Koniges discloses the use of a separate processor known as a “master processor:

Adding to the terminology confusion, we have SPMD, which stands for Single Program Multiple Data and is basically a restriction of the MIMD model. In SPMD, each processor executes exactly the same program on a different subset of the program data. Every data item in the program is allocated to one

*processor and often every processor is allocated approximately equal numbers of data items. In practice, however, even with the SPMD model, **the programmer can designate a master processor** by judicious use of conditional statements. While some of this terminology is being replaced by a smaller set of acronyms described in the next section, (e.g., MIMD-DM becomes simply MPP) it is still found and referred to both in the papers in this book and throughout the literature.*

Koniges, p. 3 (EX1019)

86. A POSITA would recognize the “master processor” of Koniges to be the “at least one further processor in addition to the subtask processors” of claim 2 of the Patent.

87. A POSITA would recognize that all of the architectures claimed in the Patent are disclosed in EX1019. In particular, Koniges discloses the subject matter of claims 2, 12, and 13 of the Patent.

L. Schwiegelshohn et al. (1998) (EX1021)

88. In 1998, Schwiegelshohn et al. published a paper entitled “Analysis of First-Come-First-Serve Parallel Job Scheduling” (EX1021), “Schwiegelshohn”). It appeared in *SODA '98: Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*.

89. It discloses:

*Today's massively parallel computers are built to execute a large number of different and independent jobs with a varying degree of parallelism. For reasons of efficiency most of these architectures allow space sharing, i.e. the concurrent execution of jobs with little parallelism on disjoint node sets. This produces a complex management task with the scheduling problem, i.e. the assignment of jobs to nodes and time slots being a central part. ... In our paper we want to demonstrate that there are algorithmic issues in job scheduling where theoretical and applied research can both contribute to a solution. **To this end we discuss First-come-first-serve (FCFS) scheduling**, a simple job scheduling method which can often be found in real systems but is usually considered inadequate by many researchers.*

EX1021, p. 629

90. A POSITA would recognize that the First-come-first serve parallel scheduling algorithm of EX*Schwie corresponds to the “first-come/first-served basis” of limitation 1(d) of the Patent,

M. Barkes et al. (1998) (EX1020)

91. In 1998, Barkes et al. published IBM Technical Report SG24-5165-00, entitled “GPFS: A Parallel File System” (EX1020, “Barkes”). It contains over 200 pages of disclosure on how to use GPFS in a variety of applications. In particular, it discloses:

5.3.3 Parallel Sorting Application

Sorting of large data sets is done frequently in database management and results in a variety of report generation tasks. In large mass-storage devices, large data files are kept sorted with respect to certain fields so that on-line queries for data can be processed very quickly. Sorting large data sets is very I/O- and data-intensive. Performance of the sort utility provided by a transaction management system forms one of the fundamental measures of processing power for the sorting mechanism.

EX1020, p. 157

92. A POSITA would recognize that EX*Barkes discloses both the sort process of claim 8 of the Patent and the report-creating process of claim 10 of the Patent.

N. **Chow et al. (2001) (EX1003)**

93. On June 27, 1997, Chow et al. filed an application that issued as U.S. Patent 6,304,866 on October 16, 2001, entitled “Aggregate job performance in a multiprocessing system by incremental and on-demand task allocation among multiple concurrently operating threads” (EX1003, “Chow”). It describes the division of a computing problem into subtasks for parallel processing:

In a multiprocessing system, multiple concurrently operating task execution units are operated to perform an aggregate task by using incremental and on-demand sub-task allocation. A command is received to perform a machine executed task divisible into multiple sub-tasks, i.e., an “aggregate task”. A

granularity is then established, for dividing the aggregate task into sub-tasks. Preferably, the granularity is not too large to permit potentially uneven sub-task allocation, and not too small to incur excessive overhead in allocating sub-tasks. Having established the granularity, multiple task execution units are independently operated on-demand to sequentially self-allocate and execute sub-tasks of the aggregate tasks.

EX1003, Abstract

94. Chow improves distributed multiprocessing systems, particularly with respect to database operations. It discloses, e.g., partitioning a task into subtasks, processing the subtasks in parallel, and merging the results to obtain a final solution:

Broadly, the present invention concerns the performance of an aggregate database processing task in a multiprocessing computer database system. The aggregate task is executed with multiple concurrently operating task execution units, using incremental and on-demand sub-task allocation. A command is received to perform a machine executed task divisible into multiple sub-tasks, i.e., an "aggregate task". Then, multiple task execution units are independently operated on-demand to sequentially self allocate and execute sub-tasks of the aggregate task. Each sub-task is sized according to a specific "granularity", used to divide the aggregate task into sub-tasks. The granularity may be predetermined for all task execution units or repeatedly dynamically determined by each task execution unit at the time of self-allocation. Preferably, the granularity is not too large as

*to cause potentially uneven sub-task allocation, and not so small
as to incur excessive overhead in allocating sub-tasks.*

EX1003, 1:61-2:11

95. This process is shown in Fig, 6 of Chow:

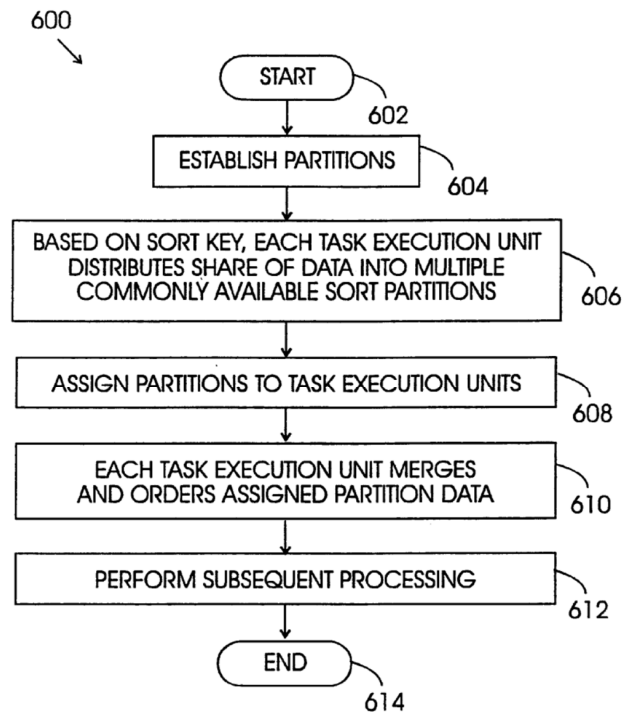


FIG. 6

96. Task execution is described in Chow as follows:

In one implementation, each task execution unit operates as follows. First, the task execution unit selects an unexecuted sub-task of the aggregate task, where the sub-task is sized according to the appropriate granularity. Then, the task execution unit executes the selected sub-task. After executing the sub-task, the task execution units determines whether any available work of the aggregate task remains. As long as available work of the

aggregate task remain, the selection and allocation steps are repeated.

EX1003, 2:21-30

97. Chow thus discloses a first-come-first-served protocol.

98. Chow is an obviousness reference applied in detail in Grounds 1 and 2 below.

O. Reiffin (2001) (EX1004)

99. On September 9, 1994, Reiffin filed an application that issued as U.S. Patent 6,330,583 on December 11, 2001, entitled “Computer network of interactive multitasking computers for parallel processing of network subtasks concurrently with local tasks” (EX1004, “Reiffin”). Like Chow, it also describes the division of a computing problem into subtasks for parallel processing:

A local area computer network provides distributed parallel processing. The network comprises a plurality of workstations or personal computers, each having preemptive multitasking for the interactive execution of a local task in the foreground concurrently with a remote network subtask in the background. A large compute-intensive task may be partitioned into a plurality of parallel subtasks executed simultaneously with each subtask executed in the background by a respective workstations without substantial interference with the local task being executed concurrently in the foreground. The computer time and processing power which would otherwise be wasted while

waiting for slow input/output operations is instead utilized to provide a powerful parallel multiprocessor system for handling compute-intensive tasks too large for an individual workstations.

EX1004, Abstract

100. Reiffin discloses a first-come-first-served (queue) protocol:

If a computer has a task which is both large and compute intensive, the task is partitioned into a plurality of smaller subtasks to be executed simultaneously in parallel by the other computers of the network. Each of these subtasks is assigned its respective identifier. One of the network disk drives has a directory constituting a pool for storing all of the subtask identifiers, preferably in the form of a queue. Each of the other computers of the network selects a subtask identifier from the pool, copies the corresponding subtask program from the originating computer, executes the selected subtask in the background while concurrently executing its own local task in the foreground, returns the computed results of the subtask computation to the originating computer, and then selects another subtask from the pool to repeat the cycle.

Reiffin, 2:19-33

101. Note that Reiffin discloses “subtask identifiers,” which correspond to the “descriptions of the partitions” in claim 1 of the Patent.

102. Reiffin is an obviousness reference applied in detail in Grounds 1 and 2 below.

P. Kurowski et al. (2011) (EX1005)

103. On January 12, 2011, Kurowski et al, filed an application that was published on February 14, 2002 as U.S. Patent Application Publication 2002/0019844, entitled “Method and system for network-distributed computing” (EX1005, “Kurowski”). It describes a large network of heterogeneous computers:

A distributed computing system achieves a highly distributed environment where very large computation intensive tasks are broken down into thousands of sub-tasks and then distributed to thousands of clients running on a variety of computers across the Internet. The idle CPU time of each of these thousands of client computers is used to perform these computations by running custom application modules in a low priority. A task server keeps track of information associated with each of the clients and uses the information to assign one or more tasks associated with a computing problem to each client computer. A file server provides the application modules to the client computers for executing their assigned tasks. An application server provides input data for the application modules and receives output data from the application modules.

Kurowski, Abstract.

104. Kurowski improves large-scale network-distributed computing systems through an architecture in which one computer doles out subtasks to collections of other computers:

FIG. 3 is a system organization diagram that illustrates the different parts that make up a system made in accordance with one exemplary embodiment of the present invention. As illustrated, client machines 200 preferably interact with file server machines 1100, task server machines 1200, and application server machines 1300. The file server machines 1100, task server machines 1200, and application server machines 1300 may be physically located at different locations and distributed across the Internet as described above. By way of example, a client 200 may receive a task to work on from a task server 1200, and the client 200 may need to get application software to work on that task from an application server 1300.

EX1005, [0061]

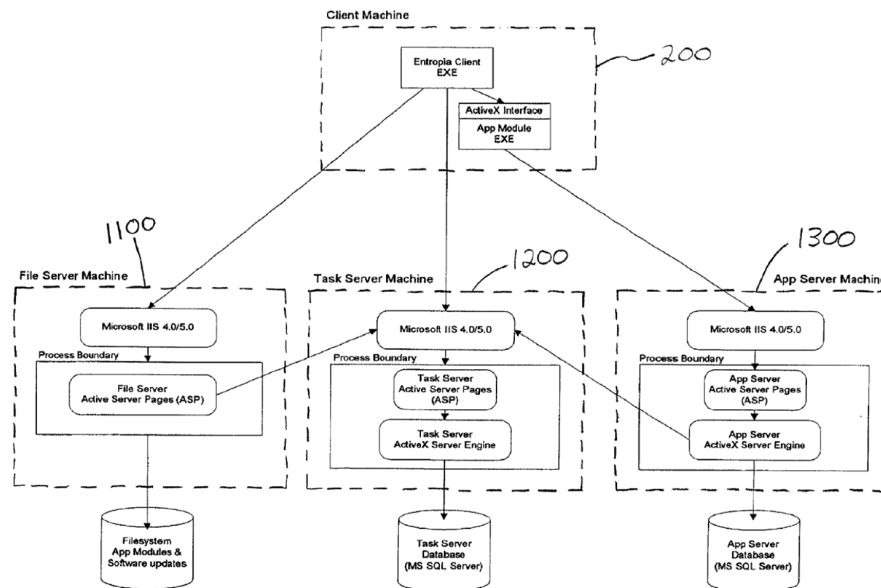


FIG. 3

EX1005, Fig. 3.

105. Kurowski is an obviousness reference applied in detail in Ground 2 below.

Q. Conclusions Regarding the Prior Art

106. Every disclosure and claim limitation in the '582 Patent was described in, and/or obvious over, one of more of the prior art items described above. None of the references in this section was considered during the '582 prosecution. There is nothing novel about the '582 Patent, which ignores thousands of prior art publications on parallel processing.

107. Owing to the long history of parallel processing, dating back at least 40 years before the '582 Patent, all of the foregoing material would have been within the general knowledge of a POSITA.

VIII. THE CHALLENGED PATENT

108. I understand that Petitioners are challenging all claims (1-14) of the '582 Patent.

109. The Patent asserts that it:

deals with a way to split the performance of a given task among a plurality of processing units which can all access, directly or indirectly, the input data and the devices on which the output data is to be stored.

Sort applications, statistical analysis batch applications and report writing applications and database queries are examples of applications that can readily enjoy this invention.

EX1001, 1:14-21

110. The Patent purports to solve these problems in a conventional manner through the following steps in claim 1¹:

111. [1pre] “A method of effecting on a preexisting input file a computer-executable process comprised of a plurality of subtasks, the method comprising the steps of:”

112. The preamble was known and conventional as far back as 1960, as I showed above in the Background of the Prior Art section in a paper by Yarbrough. EX1009. It was conventional to present a problem to a computer in the form of an input file and to break a computation into subtasks.

113. [1a] “(a) automatically determining file allocation and logically subdividing records of said input file into a plurality of partitions;” As I showed above in the Background of the Prior Art section, this step was known at least by 1960, as shown by the disclosure of Yarbrough. EX1009.

114. [1b] “distributing descriptions of all of said partitions to each of a plurality of subtask processors.” The Specification does not refer to any such “descriptions,” which are not mentioned at all except in the claims, but a POSITA would understand that such a description could comprise the location of storage

¹ All the claims of the Patent are method claims.

where data for a subtask could be found. This step was also known at least by 1960, as shown above in the Background of the Prior Art section by the disclosure of Yarbrough EX1009. Further, there are only two ways for a processor to obtain the data it needs to process a subtask, both obvious: (1) the data is transmitted to the processor; or (2) the processor is informed where it can find the data.

115. [1c] “simultaneously executing at least a respective one of the subtasks of the computer-executable process in each of at least some of said processors on a respective one of the partitions with each subtask reading and processing the respective partition so as to process the respective partition and produce respective subtask output and;” That is merely a description of what constitutes parallel processing and was known by 1960 as shown above in the Background of the Prior Art section by the disclosure of Yarbrough. Ex1009.

116. [1d] “thereafter repeating step (c) in at least some of the subtask processors each with another unprocessed partition on a first-come/first-served basis; and ...” This simply says to have a processor work on a next partition when it finishes processing its current partition. This was also commonly known by at least 1998, as shown in the Background of the Prior Art section concerning Schwiegelshohn EX1021.

117. [1e] “generating at least one output combining all of the subtask outputs and reflecting the processing of all of said subtasks.” Of course, when a problem is

divided into partitions and each partition is processed separately, the results of processing the individual partitions must be combined to yield an overall result. Otherwise the purpose of dividing into partitions would be completely defeated. As shown above in the Background of the Prior Art section, this step was known by 1960 in the disclosure of Yarbrough. EX1009.

118. Claim 2 depends from claim 1 and adds “wherein the automatic determination of file allocation and logical subdivision of records of said input file into said plurality of partitions in step (a) and the distribution of the description of all of said partitions in step (b) is carried out with at least one further processor in addition to the subtask processors.” This is precisely the role of the Monitor processor in Yarbrough’s 1960 paper, as discussed above in the Background of the Prior Art section. EX1009. See also the discussion of Koniges. Ex1019.

119. Claim 3 depends from claim 1, but adds “further comprising the step of merging said subtask outputs in step (e).” The Patent draws no distinction between “combining” subtask outputs (step [1e]) and “merging” subtask outputs (the step of claim 3). Whether not there is a difference, it was conventional to “merge” subtask outputs, in particular, when performing sorting, as discussed above in the Background of the Prior Art section with respect to Deo (EX1014), Quinn (EX1011), and Rüb (EX1016).

120. Claim 4 depends from claim 1 and adds “wherein the output in step (e) is a succession of outputs from said subtasks in a one-to-one correspondence with said records of said input file.” The meaning of this claim is unclear, as the Patent does not explain what is meant by a “succession of outputs.” As best I can determine, the claim says that each record in the input file results in one output from some subtask. This is a conventional means of processing independent input records. (Records are “independent” if processing any one of them does not affect the processing of any other, which implies that they can be processed in any order.) For example, if n records are to be processed, each separately, and m processors are available, then the conventional way of processing the records in parallel is to hand the first m records to the m processors, and, as each of the m processors finishes processing a record, then to hand that processor the next record. If processing a record takes time T , then all of the records can be processed in time Tn/m on m processors, as opposed to Time Tn on one processor, achieving the maximum speedup possible. For example, if check clearing is to be done in parallel for n checks, then each check results in either a debit or a credit to some bank account, so the debits/credits are in a one-to-one correspondence with the checks. This is a completely obvious way of processing independent records.

121. Claim 5 depends from claim 4, and adds “wherein the output in step (e) is an accumulation of output records from said subtasks in an arbitrary order.” It

was well-known in parallel processing that different processors would complete work at different times and therefore the output records would be accumulated in an unknown (arbitrary) order.

122. Claim 6 depends from claim 1, and adds “wherein said input file resides on a storage area network and is derived therefrom.” The Patent does not explain who a file can be “derived” from a storage device, so I have taken the word “derived” in this claim (and in claim 7) to mean “obtained.” Storage area networks were well-known and on the market in the mid-1990s. The use of storage area networks in parallel processing was disclosed at least by 1997 in the chapter by Blelloch et al. (EX1017) discussed in the Background of the Prior Art section.

123. Claim 7 depends from claim 1, and adds “wherein said input file resides on a network-attached storage and is derived therefrom.” Network-attached storage was well-known and commercially available in the 1980s. The use of network attached storage in parallel processing was disclosed at least by 1997 in the chapter by Blelloch et al. discussed in the Background of the Prior Art section. EX1017.

124. Claim 8 depends from claim 1 and adds “wherein said computer-executable process is a sort process.” As discussed in the Background of the Prior Art section, sorting was one of the earliest applications of parallel processing. See, specifically Barkes (EX1020), Baer (EX1010), Deo (EX1014), Quinn (EX1011), and Rüb (EX1016).

125. Claim 9 depends from claim 1 and adds “wherein said computer-executable process is a statistical analysis process.” As discussed in the Background of the Prior Art section, parallel processing was used extensively in statistical computation. See, e.g., Xu (EX1015) and Kearsley (EX1018). Further, many operations in statistical analysis, such as adding long sequences of numbers, were known to be particularly amenable to parallel processing.

126. Claim 10 depends from claim 1 and adds “wherein said computer-executable process is a report-creating process.” As discussed in the Background of the Prior Art section, it was known at least by 1998 to use parallel processing to create reports. See, e.g., Barkes (EX1020).

127. Claim 11 depends from claim 1 and adds “wherein said computer-executable process includes a database query.” As discussed in the Background of the Prior Art section, it was conventional to use parallel processing to execute database queries. See, e.g., Hu (EX1012) and Barkes (EX1020).

128. Claim 12 depends from claim 2 and adds “wherein said one processor is part of a mainframe computer and the other processors are processors of at least one other computer.” As discussed in the Background of the Prior Art section, it was known to implement parallel processing using a mainframe computer and at least one other computer. See, e.g., Hurson (EX1013) and Koniges (EX1019).

129. Claim 13 depends from claim 1 and adds “wherein said plurality of processors are all parts of a single multiprocessor.” As discussed in the Background of the Prior Art section, it was conventional to perform parallel processing on a single multiprocessor comprising a plurality of other processors. This is the very meaning of “multiprocessor.” See, e.g., Hurson (EX1013) and Koniges (EX1019).

130. Claim 14 depends from claim 1 and adds “wherein the automatic determination of file allocation and logical subdivision of records of said input file into said plurality of partitions in step (a) and the distribution of the descriptions of all of said partitions in step (b) is carried out with at least one processor, and said one processor and said plurality of processors are all parts of a single multiprocessor not including said one processor.” This claim is self-contradictory. It says that “said one processor” is part of a single microprocessor “not including said one processor. It is logically impossible for something to part of something else and yet not be included in that something else. As written, the limitation renders claim 14 inoperable, as it cannot under any circumstances be satisfied.

131. It appears to me that claim 14 attempts to claim an embodiment described at 2:33-37 (embodiment shown in boldface):

*Without limitation thereto, **the one processor can be part of a mainframe computer and the plurality of processors can be processors of at least one other computer or the plurality of processors can all be parts of a single multiprocessor.***

132. That is, claim 14 should have read, “wherein the automatic determination of file allocation and logical subdivision of records of said input file into said plurality of partitions in step (a) and the distribution of the descriptions of all of said partitions in step (b) is carried out with at least one processor, ~~and said one processor~~ and said plurality of processors are all parts of a single multiprocessor not including said one processor.” However, even if claim 14 is corrected in this manner, such an organization of parallel processors was conventional, as shown above in the disclosure of Yarbrough in the Background of the Prior Art section (EX1009).

IX. GROUND 1: CLAIMS 1-14, OBVIOUS OVER CHOW IN VIEW OF REIFFIN

A. Motivation to combine Reiffin (EX1004) with Chow (EX1003).

133. Chow and Reiffin both share the '582 Patent's objective of decomposing linear computational processes that currently use a single computer into equivalent parallel processes that can use any number of heterogeneous computers. *See, e.g.*, EX1001, 1:48-54; EX1003, 2:42-46 (“The invention affords its users with a number of distinct advantages. Chiefly, the invention guides a multiprocessing system to perform an aggregate task as quickly as possible, with minimum workload skew among independently operating processing elements.”); EX1004, 2:1-7 (“It is a further object of the invention to provide a system wherein a network of workstations or personal computers may operate in parallel to form a

powerful multicomputer system for parallel processing of large time-consuming compute-intensive applications in a fraction of the time that it would take any of the individual computers to do the job alone.”).

134. Chow is ready for improvement. Chow focuses on multiprocessing systems consisting of a single computer with multiple processors executing programs in parallel. Reiffin expands this concept to multiple networked computers. A POSITA would have recognized that benefit of combining Reiffin with Chow would be increased computational power and the ability to solve larger problems.

135. Combining Chow’s multiprocessing system with Reiffin’s teachings of a distributed parallel processing network would have been a simple combination of familiar elements to obtain a predictable result and would thus have been obvious to a POSITA.

136. It would have been routine to add the features of Reiffin to the system of Chow because the architectures of both are similar. A POSITA would thus have been motivated to combine Chow and Reiffin with a reasonable expectation of success.

B. Claim 1

137. Claim 1 as a whole and each limitation of claim 1 is disclosed or suggested by Chow and Reiffin, as shown below.

1. Preamble

[1pre] A method of effecting on a preexisting input file a computer-executable process comprised of a plurality of subtasks, the method comprising the steps of:

138. Breaking down a computer process into subtasks is part and parcel of parallel processing, the whole point of which is to divide work among multiple processors. This preamble is entirely conventional.

139. Whether or not the preamble is limiting, it is disclosed by Chow (EX1003). Chow discloses a method of effecting on a preexisting input file a computer-executable process comprised of a plurality of subtasks.

140. For example, Chow claims a “method” of “independently operating each one of multiple task execution units to sequentially and independently self-allocate and execute sub-tasks of the aggregate task on-demand.” EX1003, 10:10-13. Chow further discloses this method, noting that “the method sequence 400 is used to perform an aggregate database processing task, by breaking the task down into smaller units, and executing the units with multiple concurrently operating task execution units.” EX1003, 5:1-5.

141. The preamble is also disclosed by Reiffin: “This invention relates to computer networks, and more particularly, to a distributed parallel processing network System wherein a large compute-intensive task may be partitioned into

subtasks which are then distributed among a plurality of personal computer or workstations for parallel execution of the subtasks.” EX1004, 1:7-12.

142. Thus, Chow and Reiffin both disclose or suggest claim [1pre] (as they must to perform parallel processing).

2. Limitation 1a

[1a] (a) automatically determining file allocation and logically subdividing records of said input file into a plurality of partitions;

143. The division of an input file into partitions is part and parcel of parallel processing, which requires that a file be divided up into pieces for processing by separate processors. This limitation is therefore entirely conventional.

144. Chow (EX1003) discloses limitation [1a] (as it must to perform parallel processing). For example, Chow discloses that “[t]ypically, workload is apportioned by somehow logically dividing the data to be processed. For example, a block of data might be divided evenly into a number of parts equal to the number of available threads, so that each thread can independently process a separate portion of the data. This is called “static” or “a priori” partitioning.” Ex.1003, 1:38-42. Chow further discloses:

[a] command is received to perform a machine-executed task divisible into multiple sub-tasks, i.e., an ‘aggregate task’. Then, multiple task execution units are independently operated on-

demand to sequentially self-allocate and execute sub-tasks of the aggregate task. Each sub-task is sized according to a specific 'granularity', used to divide the aggregate task into sub-tasks. The granularity may be predetermined for all task execution units or repeatedly dynamically determined by each task execution unit at the time of self-allocation. Preferably, the granularity is not too large as to cause potentially uneven sub-task allocation, and not so small as to incur excessive overhead in allocating sub-tasks. Operating 'on-demand', each task execution unit sequentially allocates and executes one sub-task at a time, then proceeding to allocate the next unallocated [piece] of work to itself according to the appropriate granularity.

EX1003, 1:66-2:15.

As a result, Chow discloses determining file allocation and logically subdividing records of said input file into a plurality of partitions, i.e. partitions.

145. Reiffin (Ex1004) also discloses limitation [1a]. For example, Reiffin discloses that “[t]his invention relates to computer networks, and more particularly, to a distributed parallel processing network system wherein a large compute-intensive task may be partitioned into subtasks.” EX1004, 1:8-10. Reiffin further discloses:

If a computer has a task which is both large and compute-intensive, the task is partitioned into a plurality of smaller subtasks to be executed simultaneously in parallel by the other

computers of the network. Each of these subtasks is assigned its respective identifier.

EX1004, 2:19-23

[t]he workstations or personal computer originally given such a task may partition the data and program code into a plurality of subtasks. A plurality of subtask identifiers corresponding to the partitioned subtasks are stored in a network directory of a disk drive of one of the computers to constitute a pool or queue identifying remote subtasks to be executed.

EX1004, 4:64-5:3.

146. Reiffin therefore discloses determining file allocation and logically subdividing records of said input file into a plurality of partitions.

147. Thus, Chow and Reiffin both separately disclose or suggest claim limitation [1a] (as they must to perform parallel processing).

3. Limitation 1b

[1b] distributing descriptions of all of said partitions to each of a plurality of subtask processors

148. As mentioned in ¶ 114, the Patent provides no explanation of what a “description” might comprise. A POSITA would understand that pointing to a storage location where the subtask processor can find the data it is to process would constitute the claimed “description.”

149. Chow (EX1003) teaches or suggests limitation [1b]. Chow discloses that partitions are accessible by all subtask processors. For example, Chow discloses that “[t]he common partition is accessible by all task execution units 101-104.” EX1003, 9:37-38. As such, each of the task execution units have access to information about the partition.” This “information” corresponds to the claimed “descriptions.” Chow further discloses: “[a] command is received to perform a machine-executed task divisible into multiple sub-tasks, i.e., an ‘aggregate task’. Then, multiple task execution units are independently operated on-demand to sequentially self-allocate and execute sub-tasks of the aggregate task.” EX1003, 1:66-2:15.

150. Reiffin (EX1004) also teaches or suggests claim [1b]. Reiffin discloses that “[i]f a computer has a task which is both large and compute-intensive, the task is partitioned into a plurality of smaller subtasks to be executed simultaneously in parallel by the other computers of the network. Each of these subtasks is assigned its respective identifier. One of the network disk drives has a directory constituting a pool for storing all of the subtask identifiers, preferably in the form of a queue.” EX1004, 2:19-25. Reiffin therefore discloses providing descriptions (identifiers) of the partitions to each of the subtask processors.

151. Thus, Chow and Reiffin both separately disclose or suggest limitation [1b].

4. **Limitation 1c**

[1c] (c) simultaneously executing at least a respective one of the subtasks of the computer-executable process in each of at least some of said processors on a respective one of the partitions with each subtask reading and processing the respective partition so as to process the respective partition and produce respective subtask output and;

152. This step is not only a conventional step in parallel processing – it is the very essence of parallel processing and is as old as parallel processing itself.

153. Chow discloses that “[g]enerally, the method sequence 400 is used to perform an aggregate database processing task, by breaking the task down into smaller units, and executing the units with multiple concurrently operating task execution units.” EX1003, 5:1-5. A POSITA would understand Chow’s “units” to be the claimed “partitions.” Chow describes the concurrent execution of subtasks to produce a resulting output: “After all work is performed, results are pooled for possible final processing steps and to present as a final output.” EX1003, 7:46-48.

154. Reiffin discloses that “a large compute-intensive task may be partitioned into subtasks which are then distributed among a plurality of personal computer or workstations for parallel execution of the subtasks.” EX1004, 1:9-12. Reiffin further discloses that “[e]ach of the other computers of the network selects a subtask identifier from the pool, copies the corresponding subtask program from

the originating computer, executes the selected subtask in the background while concurrently executing its own local task in the foreground, returns the computed results of the subtask computation to the originating computer, and then selects another subtask from the pool to repeat the cycle.” Ex.1004, 2:25-33. Reiffin thus discloses the parallel execution of subtasks from a partition with each subtask processor creating results.

155. Therefore, Chow and Reiffin both separately disclose or suggest limitation [1c].

5. Limitation 1d

(d) thereafter repeating step (c) in at least some of the subtask processors each with another unprocessed partition on a first-come/first-served basis; and;

156. This is a conventional step in parallel processing. When one processor finishes its subtask, it is assigned the next unprocessed subtask. See, *e.g.*, the discussion of Schwiegelshohn beginning at ¶ 88.

157. Chow discloses that “[a]fter step 404, step 408 concurrently begins processing the aggregate task by concurrently starting each of the task execution units 101-104 on different sub-tasks. When a task execution unit finishes its sub-task, it embarks on another sub-task, if any remain. Thus, the multiprocessing system 100 functions like a “straw model”, with several people drinking water

from a common glass through individual straws.” EX1003, 5:13-21. Chow’s disclosed straw method discloses that when a subtask processor completes a subtask, it selects a new subtask for processing, and this “straw model” is performed on a first-come/first-served basis.

158. Reiffin discloses that once a subtask processor completes a task, it returns the result and then selects another subtask to process: “[i]f a computer has a task which is both large and compute-intensive, the task is partitioned into a plurality of smaller subtasks to be executed simultaneously in parallel by the other computers of the network. Each of these subtasks is assigned its respective identifier. One of the network disk drives has a directory constituting a pool for storing all of the subtask identifiers, preferably in the form of a queue.” EX1004, 2:19-25. Reiffin’s “queue” is a first-come-first-served line.

159. Thus, Chow and Reiffin both separately disclose or suggest limitation [1d].

6. Limitation 1e

(e) generating at least one output combining all of the subtask outputs and reflecting the processing of all of said subtasks.

160. This is a conventional step in parallel processing because the final answer is always obtained by combining the results of separate computations. See, e.g., the discussion of Hu at ¶ 61.

161. Chow discloses that “[i]f step 510 finds that further processing still remains, the intermediate results are again pooled in step 506, and step 508 repeated. Steps 506 and 508 are repeated until processing of the data structure is completed. At this point, task 511 pools the results for final output, then, the routine 500 ends in step 512.” EX1003, 7:66-8:4. Chow therefore discloses that the results are pooled (merged) into a final output, reflecting the processing of all the subtasks.

162. Reiffin discloses that “[a]s explained in more detail below, each computer of the network may access this pool, select a subtask identifier, copy the corresponding remote subtask from the originating computer, execute the subtask in the background without substantial delay of its local processing, and return the computed subtask results to the originating computer.” EX1004, 5:3-8. The purpose of returning the computer subtask results to the originating computer is to obtain the final result by combining (merging) the subtask results.

163. Reiffin discloses that “The computed results from the execution of each subtask are placed into files which are stored in a network directory which may be accessed by the workstations which originally partitioned and distributed the compute-intensive task.” EX1004, 6:9-13. Reiffin then also discloses generating output results, reflecting the processing of the subtasks, and storing them in a central location.

164. Thus, Chow and Reiffin both separately disclose or suggest limitation [1e].

165. Therefore, claim 1 was obvious over Chow and Reiffin separately and over their combination.

C. Claim 2

The method defined in claim 1 wherein the automatic determination of file allocation and logical subdivision of records of said input file into said plurality of partitions in step (a) and the distribution of the description of all of said partitions in step (b) is carried out with at least one further processor in addition to the subtask processors.

166. As shown in connection with claim 1, Chow and Reiffin both disclose or suggest the method defined in claim 1.

167. The additional limitation of claim 2 is a conventional step in parallel processing. It basically requires that the computer that assigned the subtasks is different from the computers that perform the subtask computations. This was disclosed by 1960 as the “Monitor processor” of Yarbrough. See, *e.g.*, ¶¶ 52-53, above. It was also disclosed as the “master processor” of Koniges in 1977. See, *e.g.*, ¶¶ 85-86, above.

168. Chow discloses that “[t]he present invention relates to multiprocessing computer database systems. More particularly, the invention concerns a method,

apparatus, and article of manufacture for performing an aggregate database processing task with multiple concurrently operating task execution units, using incremental and on-demand sub-task allocation.” EX1003, 1:10-15. Therefore, Chow discloses that the subtasks are processed by separate processors. In addition, Chow discloses yet another separate processor for partitioning: “Typically, workload is apportioned by somehow logically dividing the data to be processed. For example, a block of data might be divided evenly into a number of parts equal to the number of available threads, so that each thread can independently process a separate portion of the data. This is called “static” or “a priori” partitioning.” EX1003, 1:38-42. Chow therefore discloses execution units, *i.e.*, subtask processors, that are independent from the computer that creates the partitions.

169. Reiffin discloses that a computer, different from the subtask processors, partitions the task into subtasks to be executed by “other computers on the network”: “If a computer has a task which is both large and compute-intensive, the task is partitioned into a plurality of smaller subtasks to be executed simultaneously in parallel by the **other** computers of the network.” EX1004, 2:19-22).

170. Thus, Chow and Reiffin both disclose or suggest claim 2.

171. Therefore, claim 2 was obvious over Chow and Reiffin separately or in combination.

D. Claim 3

The method defined in claim 1, further comprising the step of merging said subtask outputs in step (e).

172. Chow and Reiffin both disclose or suggest the method of claim 1, as discussed above.

173. The merging of subtask outputs is purely conventional in parallel processing. Merging is disclosed in Quinn (see, e.g., EX1009 ¶¶ 57-58), Deo (see, e.g., EX1014 ¶¶ 69-70), and Rüb (see, e.g., EX1016 ¶¶ 74-75).

174. As stated with regard to claim [1e], Chow discloses that the results are pooled into a final output, reflecting the processing of all the subtasks. This “pooling” is a merge step. See limitation [1e], above; EX1003, 7:66-8:4.

175. And as further noted regarding claim [1e], Reiffin discloses generating output results that reflect the processing of the subtasks, and storing them in a central location so they can be combined (merged) to form a final result of parallel processing. See limitation [1e], above; EX1004, 6:9-13.

176. Thus, Chow and Reiffin both disclose or suggest claim 3. Therefore, claim 3 was obvious over Chow and Reiffin separately or in combination.

E. Claim 4

The method defined in claim 1 wherein the output in step (e) is a succession of outputs from said subtasks in a one-to-one correspondence with said records of said input file.

177. Chow and Reiffin both disclose or suggest the method of claim 1, as discussed above.

178. The additional limitation of claim 4 was conventional, as explained above in ¶ 120. In a sorting application, for example, the output records are in a one-to-one correspondence with the input records.

179. As discussed with regard to [1e], Chow discloses that the output, reflecting the processing of all the subtasks, is a succession of outputs from subtasks that relate directly to the partitions of the input file. See limitation [1e]; EX1003, 7:66-8:4. If one subtask is dedicated to each input record, the output records are in a one-to-one correspondence.

180. As further noted regarding claim [1e], Reiffin discloses generating output results reflecting the processing of the subtasks, and storing them in a central location where they relate directly to portions of the input file. See *infra* claims [1e]; EX1004, 6:9-13. Once again, if one subtask is dedicated to each input record, the output records are in a one-to-one correspondence.

181. Thus, Chow and Reiffin both disclose or suggest claim 4.

182. Therefore, claim 4 was obvious over Chow and Reiffin separately or in combination.

F. Claim 5

The method defined in claim 1 wherein the output in step (e) is an accumulation of output records from said subtasks in an arbitrary order.

183. Chow and Reiffin both disclose or suggest the method of claim 1, as discussed above.

184. As discussed in ¶ 121, above, independent records can be processed in an arbitrary order. (This is one of the properties that makes processing independent records so amenable to parallel processing.) Therefore, the limitation in claim 5 would be met by any task whose records are independent, such as check processing, fulfilling online purchase orders, and adding or multiplying a long list of numbers.

185. Chow discloses:

Such apportionment of work is rarely completely balanced. One reason is that the extent of the data may be difficult to ascertain, or the data's processing time may be otherwise difficult to predict. Furthermore, even equisized data subparts may require comparatively different processing times. As a result of this lack of balance, some threads finish early, and other threads labor for longer periods. Still, the overall job is not complete until all

threads have finished, including the last to finish its respective task. Thus, even though some threads complete their work early, other threads linger on, delaying completion of the ultimate task. One name for this phenomenon is called 'workload skew'. For users seeking faster execution of their multi-thread application programs, workload skew is a significant problem area.

EX1003, 1:43-57

186. Because the completion time of subtasks cannot be predicted, their outputs are produced in arbitrary order.

187. And as further noted regarding claim [1e], Reiffin discloses generating output results, reflecting the processing of the subtasks, and storing, i.e., accumulating, them in a central location where they relate directly to portions of the input file. See limitation [1e], above; EX1004, 6:9-13. Reiffin discloses a heterogeneous collection of processors, which means the execution times of the subtasks will vary, implying that the output will be produced in arbitrary order.

188. Neither the Chow nor the Reiffin reference discloses a preference for whether the accumulation of the output is ordered or arbitrary. As a result, a POSITA would understand that the Chow and Reiffin references could accumulate the results in either an ordered or arbitrary way.

189. Thus, Chow and Reiffin both disclose or suggest claim 5.

190. Therefore, claim 5 was obvious over Chow and Reiffin separately or in combination.

G. Claim 6

The method defined in claim 1 wherein said input file resides on a storage area network and is derived therefrom.

191. Chow and Reiffin both disclose or suggest the method of claim 1, as discussed above.

192. The use of storage-area networks for parallel processing was known by 1998. *See* ¶ 122, *supra*.

193. Chow discloses a multiprocessing system that includes a data storage unit and a data storage facility: “The multiprocessing system 100 includes multiple task execution units 101-104, each coupled to a data storage unit 106. . . . The data storage unit 106 comprises one or more devices for electronically storing data, such as an integrated circuit memory, magnetic hard drive, optical disk drive, magnetic or optical tape, etc. As described below, the task execution units 101-104 may advantageously be used for transferring data from the data storage unit 106 to another data storage facility, as well as performing optional manipulation of the data in transit. . . . Nonetheless, depending upon the needs of the application, the memory 108 may comprise any device for electronically storing data, such as an integrated circuit memory, magnetic hard drive, optical disk drive, magnetic or

optical tape, etc.” EX1003, 3:23-49. As storage area networks were common in 2002, a POSITA would have understood a storage area network to be among Chow’s “any device for electronically storing data.

194. Reiffin also discloses uses of storage-area networks relating to the input file and subtasks: “If a computer has a task which is both large and compute-intensive, the task is partitioned into a plurality of smaller subtasks to be executed simultaneously in parallel by the other computers of the network. Each of these subtasks is assigned its respective identifier. One of the network disk drives has a directory constituting a pool for storing all of the subtask identifiers, preferably in the form of a queue.” EX1004, 2:19-25.

195. Thus, Chow and Reiffin both disclose or suggest claim 6.

196. Therefore, claim 6 was obvious over Chow and Reiffin separately or in combination.

H. Claim 7

The method defined in claim 1 wherein said input file resides on a network-attached storage and is derived therefrom.

197. Chow and Reiffin both disclose or suggest the method of claim 1, as discussed above.

198. The use of network-attached storage for parallel processing was known by 1998. *See* ¶ 122, *supra*.

199. As discussed in connection with claim 6, Chow discloses “any device for electronically storing data.” As network-attached storage was common in 2002, a POSITA would have understood network-attached storage to be among Chow’s disclosed storage devices.

200. Thus, Chow discloses or suggest claim 7.

201. Therefore, claim 7 was obvious over Chow or Chow and Reiffin in combination.

I. Claim 8

The method defined in claim 1 wherein said computer-executable process is a sort process.

202. Chow and Reiffin both disclose or suggest the method of claim 1, as discussed above.

203. This limitation was obvious on its face, as sorting was one of the earliest applications of parallel processing, and entire papers had been devoted to it. See the discussion above of Baer (EX1010), Quinn (EX1011), Deo (EX1014), Rüb (EX1016), and Barkes (EX1020). A very old method of sorting known as “merge sort” is particularly amenable to parallel processing.

204. A very old method of sorting known as “merge sort” is particularly amenable to parallel processing.

205. Chow discloses using its disclosed invention in the context of a sort process: “One mechanism to pool and redistribute data during multi-level processing is the partition sort technique, referred to as “p-sort”. Broadly, the p-sort approach divides the data into sorted partitions according to a predetermined criteria, then assigns each partition to a task execution unit.” EX1003, 8:9-13.

206. Reiffin explains that the disclosed invention can be applied to any of a variety of “large compute-intensive applications which frequently arise in science, engineering, financial analysis, image processing and other fields.” EX1004, 1:52-54. A POSITA would understand that Reiffin’s expansive potential use in any of a variety of applications would extend to include a sort process.

207. Thus, Chow and Reiffin both disclose or suggest claim 8.

208. Therefore, claim 8 was obvious over Chow and Reiffin separately or in combination.

J. Claim 9

The method defined in claim 1 wherein said computer-executable process is a statistical analysis process.

209. Chow and Reiffin both disclose or suggest the method of claim 1, as discussed above.

210. Statistical analysis was one of the early applications of parallel processing. See the discussion above of Xu (EX1015) and Kearsley (EX1018). See also ¶ 125.

211. Chow discloses that “By any name, engineers and scientists are constantly seeking new ways to minimize their applications task completion time.” EX1003, 1:23-25. A POSITA would recognize that statistical analysis is an “application” of interest to engineers and scientists.

212. Reiffin explains that the disclosed invention can be applied to any of a variety of “large compute-intensive applications which frequently arise in science, engineering, financial analysis, image processing and other fields.” EX1004, 1:52-54. A POSITA would understand Reiffin’s expansive potential use in any of a variety of applications would extend to include a statistical analysis process.

213. Thus, Chow and Reiffin both disclose or suggest claim 9.

214. Therefore, claim 9 was obvious over Chow and Reiffin separately or in combination.

K. Claim 10

The method defined in claim 1 wherein said computer-executable process is a report-creating process.

215. Chow and Reiffin both disclose or suggest the method of claim 1, as discussed above.

216. It was well-known to use parallel processing in report creation, especially if generating the contents of the report would benefit from parallel processing. See, e.g., the discussion of Barkes (EX1020) and ¶ 126.

217. Chow discloses that “By any name, engineers and scientists are constantly seeking new ways to minimize their applications task completion time.” EX1003, 1:23-25. A POSITA would recognize that report generation is an “application” of interest to engineers and scientists.

218. Reiffin explains that the disclosed invention can be applied to any of a variety of “large compute-intensive applications which frequently arise in science, engineering, financial analysis, image processing and other fields.” EX1004, 1:52-54. A POSITA would understand Reiffin’s expansive potential use in any of a variety of applications would extend to include a report generation process.

219. Thus, Chow and Reiffin both disclose or suggest claim 10.

220. Therefore, claim 10 was obvious over Chow and Reiffin separately or in combination.

L. Claim 11

The method defined in claim 1 wherein said computer-executable process includes a database query.

221. Chow and Reiffin both disclose or suggest the method of claim 1, as discussed above.

222. It was well-known to use parallel processing for database operations. For example, Hu (EX1012), discussed above in ¶¶ 59-64, was completely devoted to that topic in 1989. Hu also cites 18 prior papers on parallel database processing.

223. Chow is devoted to multiprocessor database systems:

The present invention relates to multiprocessing computer database systems. More particularly, the invention concerns a method, apparatus, and article of manufacture for performing an aggregate database processing task with multiple concurrently operating task execution units, using incremental and on-demand sub-task allocation.

EX1003, 1:10-16.

224. Chow discloses use of a “Table Scan” operation that can be used to perform database query processing steps: “The “Table Scan” operation moves rows of tables from one storage location to another. Table Scan may be used, for example, to retrieve table data from nonvolatile storage to RAM. This retrieval provides a process or “thread” with access to the data, so that the process can perform various operations on the data. These operations, for example, may involve sort, group, join, and other database query processing steps. Table Scan may be performed with granularity of one row, multiple rows, all rows on one page, or another appropriate unit of the underlying table data.” EX1003, 6:49-59.

225. Reiffin explains that the disclosed invention can be applied to any of a variety of “large compute-intensive applications which frequently arise in science, engineering, financial analysis, image processing and other fields.” Ex.1004 (1:52-54). A POSITA would understand Reiffin’s expansive potential use in any of a variety of applications, particularly financial analysis, would extend to include a database query.

226. Thus, Chow and Reiffin both disclose or suggest claim 11.

227. Therefore, claim 11 was obvious over Chow and Reiffin separately or in combination.

M. Claim 12

The method defined in claim 2 wherein said one processor is part of a mainframe computer and the other processors are processors of at least one other computer.

228. Chow and Reiffin both disclose or suggest the method of claim 2, as discussed above.

229. The earliest parallel processing computers were mainframes for the simple reason that no other forms of computers existed for a considerable period of time. Mainframes are mentioned expressly in Hurson (EX1013). Further, numerous references disclose parallel processing using more than one computer,

which was a common form of multiprocessing. See, e.g., Baer (EX1010), Hu (EX1012), Koniges (EX1019), Barkes (EX1020), and Schwiegelshohn (EX1021).

230. Chow explains that the disclosed invention is intended to reduce task complete time in multiprocessing environments, which it describes as “computer systems where multiple computers, digital data processing units, threads, or other processes perform concurrent tasks.” Ex.1003 (1:27-29). Chow further explains that “[e]ach task execution unit may also comprise a hardware device such as a computer, microprocessor, programmed memory, discrete circuitry, application-specific integrated circuitry, programmable circuit components, logic circuitry, a combination of the foregoing, or another suitable digital data processing machine.” Ex.1003 (3:58-63). Chow therefore discloses that a multi-processing system can include multiple computers, with no limitation on their potential architecture. Therefore, a POSITA would understand that Chow could be implemented in an environment in which the multiple computers include a mainframe and other computers.

231. An objective of Reiffin is to create a network of a plurality of computers for parallel processing:

It is a further object of the invention to provide a system wherein a network of workstations or personal computers may operate in parallel to form a powerful multicomputer system for parallel processing of large time-consuming compute-intensive

applications in a fraction of the time that it would take any of the individual computers to do the job alone.

EX1004, 2:1-7

232. Reiffin thus discloses a multi-computer architecture. A POSITA would realize that a mainframe, being even more capable than a workstation, could be employed in place of one of the workstations.

233. Thus, Chow and Reiffin both disclose or suggest claim 12.

234. Therefore, claim 12 was obvious over Chow and Reiffin separately or in combination.

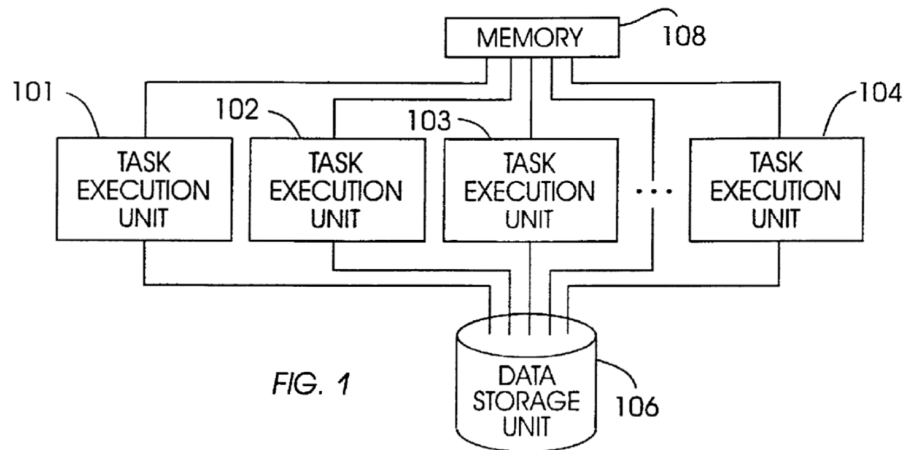
N. Claim 13

The method defined in claim 1 wherein said plurality of processors are all parts of a single multiprocessor.

235. Chow and Reiffin both disclose or suggest the method of claim 1, as discussed above.

236. Parallel processing on a single multiprocessor was disclosed at least by 1960 in Yarbrough (EX1009). It is also disclosed in Hurson (EX1013) and Koniges (EX1019).

237. Fig. 1 of Chow discloses an architecture with multiple task execution units sharing a common memory and a common data storage unit:



238. A POSITA would recognize that the task execution units could all be processors of a single computer. In fact, such an organization would be natural in view of the shared memory and data storage unit.

239. Thus, Chow discloses or suggests claim 13.

240. Therefore, claim 13 was obvious over Chow or Chow in combination with Reiffin.

O. Claim 14

The method defined in claim 1 wherein the automatic determination of file allocation and logical subdivision of records of said input file into said plurality of partitions in step (a) and the distribution of the descriptions of all of said partitions in step (b) is carried out with at least one processor, and said one processor and said plurality of processors are all parts of a single multiprocessor not including said one processor.

241. Chow and Reiffin both disclose or suggest the method of claim 1, as discussed above.

242. I note in ¶¶ 130-132 that claim 14 is inoperable as written and incapable of enablement because it is logically contradictory. As such, it would be impossible to locate any prior art over which an inoperable claim would be obvious.

243. I believe the claim should have read, “The method defined in claim 1 wherein the automatic determination of file allocation and logical subdivision of records of said input file into said plurality of partitions in step (a) and the distribution of the descriptions of all of said partitions in step (b) is carried out with at least one processor, and said plurality of processors are all parts of a single multiprocessor not including said one processor.” With that correction, the claim is obvious, and was disclosed in Yarbough in 1960, which distinguishes a “monitor processor” from the other processors. EX1009. It is also disclosed in Koniges, which refers to a “master processor.” EX1019.

244. It therefore would have been obvious to a POSITA to use one of Chow’s task execution units as a “monitor” or “master” processor.

245. Thus, Chow discloses or suggests claim 14.

246. Therefore, claim 14 was obvious over Chow and the combination of Chow and Reiffin.

X. GROUND 2: CLAIMS 1-4 WERE OBVIOUS OVER CHOW IN VIEW OF REIFFIN, FURTHER IN VIEW OF KUROWSKI

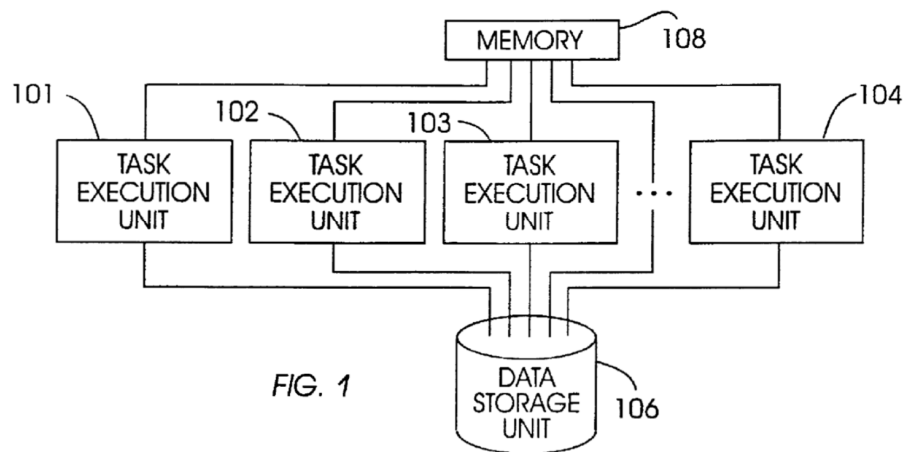
247. Every limitation of claims 1-14 is disclosed or suggested by Chow (EX1003), Reiffin (EX1004), and Kurowski (EX1005).

A. Motivation to Combine

248. For all the reasons previously stated it would have been simple to combine Chow's multi-processing system with Reiffin's teachings of a distributed parallel processing network resulting in the combination of known elements to obtain a predictable result, and thus such combination would be obvious to a POSITA. In addition, Kurowski is also directed to the '582 Patent's goal of decomposing linear processes that currently use a single computer into equivalent parallel processes that can use any number of heterogeneous computers. For example, "A distributed computing system achieves a highly distributed environment where very large computation intensive tasks are broken down into thousands of sub-tasks and then distributed to thousands of clients running on a variety of computers across the Internet." EX1005, Abstract.

249. Both Chow and Reiffin were ready for improvement. Chow focuses on multiprocessor systems consisting of multiple processors. Reiffin focuses on multiprocessor systems consisting of multiple separate computers connected via a local area network. Kurowski expands the concept to "thousands of clients running on a variety of computers across the Internet." *Id.*

250. It would have been routine to add the features of Reiffin to the system of Chow, as previously discussed. It would have also been routine to add the features of Kurowski to Chow and Reiffin. In fact, the architectures of Chow and Kurowski are very similar in terms of task breakdown. *Cf.*, Fig. 1 of Chow and Fig. 3 of Kurowski:



EX1003, Fig. 1

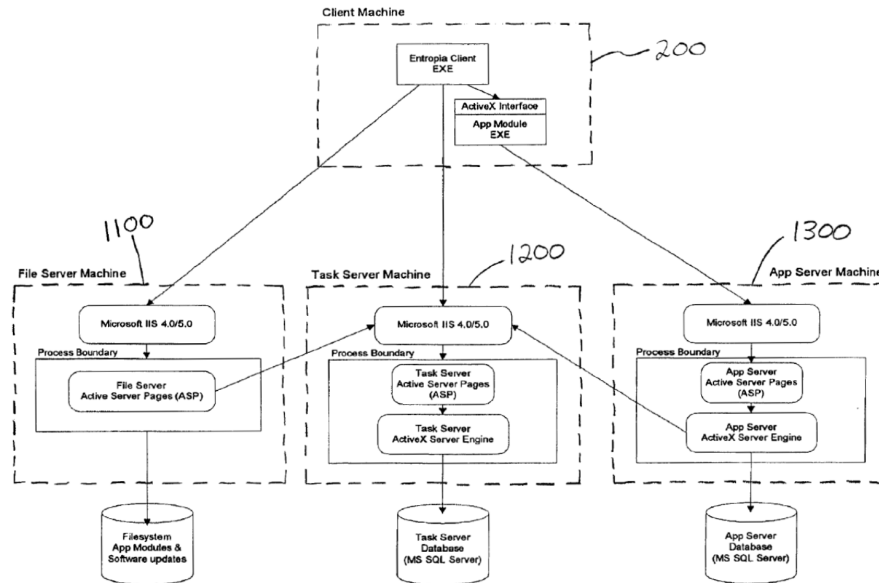


FIG. 3

EX1005, Fig. 3.

251. It was well-known as of the Priority Date how to interconnect heterogeneous computers over the Internet. This was one of the reasons the Internet was developed.

252. A POSITA would have thus been motivated to combine Chow, Reiffin, and Kurowski with a reasonable expectation of success. Claims 1-14 are therefore obvious over Chow in view of Reiffin and further in view of Kurowski.

B. Claim 1

1. Limitation [1pre]

A method of effecting on a preexisting input file a computer-executable process comprised of a plurality of subtasks, the method comprising the steps of:

253. As previously noted in Ground 1, Chow and Reiffin each discloses or suggests the preamble [1pre].

254. Breaking down a computer process into subtasks is part and parcel of parallel processing, the whole point of which is to divide work among multiple processors. This preamble is entirely conventional.

255. Kurowski also discloses this preamble. It discloses a method of effecting on a preexisting input file a computer-executable process comprised of a plurality of subtasks. For example, Kurowski explains:

[t]he client 200 is one of the pieces in the larger picture of how the distributed computing system 100 described herein provides a highly distributed environment for large scale computing problems. One goal of all these different pieces is to enable the system to take on a very large computing problem, break it down into thousands of tasks and distribute it to tens of thousands of computers across the internet so this large computation could be performed by tens of thousands of parallel sub-computations and the results brought back from all these different clients 200.

EX1005, [0192]

256. Thus, Chow, Reiffin, and Kurowski, separately and together disclose or suggest limitation [1pre].

2. Limitation [1a]

**(a) automatically determining file allocation and logically
subdividing records of said input file into a plurality of partitions;**

257. As previously noted in Ground 1, each of Chow and Reiffin discloses or suggests limitation [1a].

258. The division of an input file into partitions is part and parcel of parallel processing, which requires that a file be divided up into pieces for processing by separate processors. This limitation is therefore entirely conventional.

259. Kurowski discloses limitation [1a]. For example, Kurowski discloses that:

[t]he client 200 is one of the pieces in the larger picture of how the distributed computing system 100 described herein provides a highly distributed environment for large scale computing problems. One goal of all these different pieces is to enable the system to take on a very large computing problem, break it down into thousands of tasks and distribute it to tens of thousands of computers across the internet so this large computation could be performed by tens of thousands of parallel sub-computations and the results brought back from all these different clients 200.

EX1005, [0192]

Kurowski, therefore, discloses automatically identifying a large computing task and partitioning it.

260. Thus, Chow, Reiffin, and Kurowski each discloses or suggests limitation [1a].

3. Limitation [1b]

[1b] (b) distributing descriptions of all of said partitions to each of a plurality of subtask processors

261. As mentioned in ¶ 114, the Patent provides no explanation of what a “description” might comprise. A POSITA would understand that pointing to a storage location where the subtask processor can find the data it is to process would constitute the claimed “description.”

262. As previously noted in Ground 1, Chow and Reiffin each discloses or suggests limitation [1b].

263. Kurowski teaches or suggests claim [1b]. Kurowski discloses that “[s]pecifically, all machine, user, team, and task information, summaries of such information, or subsets of such information, may be made available through reports and interactive pages on an “integrated” web site.” EX1005, [0188]. As such, partition descriptions (“such information”) are distributed and made available to all of the processors.

264. Thus, Chow, Reiffin, and Kurowski each discloses or suggests limitation [1b].

4. Limitation [1c]

[1c] (c) simultaneously executing at least a respective one of the subtasks of the computer-executable process in each of at least some of said processors on a respective one of the partitions with each subtask reading and processing the respective partition so as to process the respective partition and produce respective subtask output and;

265. As previously noted in Ground 1, Chow and Reiffin each disclose or suggests limitation [1c].

266. This step is not only a conventional step in parallel processing – it is the very essence of parallel processing and is as old as parallel processing itself.

267. Kurowski discloses that “[w]ith this system, one client 200 can work on one part of a calculation while a different client 200 works on another part of the same calculation. The distributed network of “servers” make the calculation all come together transparently.” EX1005, [0043]. Kurowski further explains that “[o]ne goal of all these different pieces is to enable the system to take on a very large computing problem, break it down into thousands of tasks and distribute it to tens of thousands of computers across the internet so this large computation could be performed by tens of thousands of parallel sub-computations and the results brought back from all these different clients 200.” EX1005, [0192]. The

breakdown is into “partitions,” as claimed. Kurowski describes concurrent execution of subtasks to produce a resulting output.

268. Thus, Chow, Reiffin, and Kurowski each discloses or suggests claim [1c].

5. Limitation [1d]

[1d] (d) thereafter repeating step (c) in at least some of the subtask processors each with another unprocessed partition on a first-come/first-served basis; and

269. As previously noted in Ground 1, Chow and Reiffin each discloses or suggests limitation [1d].

270. This is a conventional step in parallel processing. When one processor finishes its subtask, it is assigned the next unprocessed subtask. See, *e.g.*, the discussion of Schwiegelshohn beginning at ¶ 88. EX1021.

271. Kurowski discloses that “[p]referably, the client 200 requests work rather than being given a task. In other words, the client 200 preferably requests or asks for work rather than receiving a start message from a server.” EX1005, [0069]. Kurowski further explains that “the client 200 preferably requests tasks from the Task Server 1200. At appropriate times in the client’s execution, it may decide that its current task is finished and it needs to obtain a new task from the Task Server 1200. . . . In return, in step 222 the Task Server 1200 assigns a task to

the client 200 based on information received from the client 200.” EX1005, [0154].

As a result, Kurowski discloses a system in which subtasks are selected and processed on a first-come/first-served basis.

272. Thus, Chow, Reiffin, and Kurowski each discloses or suggests claim [1d].

6. Limitation [1e]

[1e] (e) generating at least one output combining all of the subtask outputs and reflecting the processing of all of said subtasks

273. As previously noted in Ground 1, each of Chow and Reiffin discloses or suggests claim [1e].

274. This is a conventional step in parallel processing because the final answer is always obtained by combining the results of separate computations. See, e.g., the discussion of Hu at ¶ 61.

275. Kurowski discloses that “[w]ith this system, one client 200 can work on one part of a calculation while a different client 200 works on another part of the same calculation. The distributed network of “servers” make the calculation all come together transparently.” EX1005, 6:9-13. Kurowski further explains that “[o]ne goal of all these different pieces is to enable the system to take on a very large computing problem, break it down into thousands of tasks and distribute it to tens of thousands of computers across the internet so this large computation could

be performed by tens of thousands of parallel sub-computations and the results brought back from all these different clients 200.” EX1005, [0192]. Kurowski therefore discloses that the results are pooled (combined) into a final result.

276. Thus, Chow, Reiffin, and Kurowski each discloses or suggests limitation [1e].

7. **Claim 2:**

2. The method defined in claim 1 wherein the automatic determination of file allocation and logical subdivision of records of said input file into said plurality of partitions in step (a) and the distribution of the description of all of said partitions in step (b) is carried out with at least one further processor in addition to the subtask processors.

277. As previously noted in Ground 1, Chow and Reiffin each discloses or suggests claim 2.

278. Chow, Reiffin, and Kurowski each discloses or suggests the method defined in claim 1.

279. The additional limitation of claim 2 is a conventional step in parallel processing. It basically requires that the computer that assigned the subtasks is different from the computers that perform the subtask computations. This was disclosed by 1960 as the “Monitor processor” of Yarbrough. See, *e.g.*, EX1009 ¶¶

52-53, above. It was also disclosed as the “master processor” of Koniges in 1977.

See, e.g., EX1019 ¶¶ 85-86, above.

280. As shown above, Chow, Reiffin, and Kurowski each discloses or suggests the method defined in claim 1.

281. Kurowski discloses a partitioning and distribution system that is separate from the subtask processors:

The present invention advantageously addresses the needs above as well as other needs by providing a system for use in distributed computing. The system includes a task server, a file server, and an application server. The task server is configured to keep track of information associated with each of a multiplicity of client computers and to use the information to assign one or more tasks associated with a computing problem to each client computer. The file server is configured to provide application modules to the client computers for executing their assigned tasks. The application server is configured to provide input data for the application modules to the client computers and to receive output data of the application modules from the client computers.

EX1005, [0008].

282. Thus, Chow, Reiffin, and Kurowski each discloses or suggests claim 2.

8. Claims 3-14

283. As shown above, Chow, Reiffin, and Kurowski each discloses or suggests the method defined in claim 1 and the method defined in claim 2.

284. Further, as previously noted in Ground 1, Chow and Chow in combination with Reiffin each discloses or suggests claims 3-14.

285. Therefore, Chow, the combination of Chow and Reiffin, and the combination of Chow and Reiffin and Kurowski discloses or suggests claims 3-14.

XI. CONCLUDING STATEMENT

286. Relying on my education, experience, skill, and training in the relevant field of art, and after reviewing the materials identified above, it is my opinion that all of the Challenged Claims are invalid for obviousness.

XII. JURAT

287. In signing this declaration, I recognize that the declaration will be filed as evidence in a contested case before the Patent Trial and Appeal Board of the United States Patent and Trademark Office. I also recognize that I may be subject to cross-examination in the case and that cross-examination will take place within the United States. If cross-examination is required of me, I will appear for cross-examination remotely within the United States during the time allotted for such cross-examination at a time and place agreed by counsel.

288. I declare 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

Declaration of Michael I. Shamos
U.S. Patent 7,257,582

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.

Executed April 2, 2025 at Pittsburgh, PA

By: Michael Ian Shamos
Michael Ian Shamos