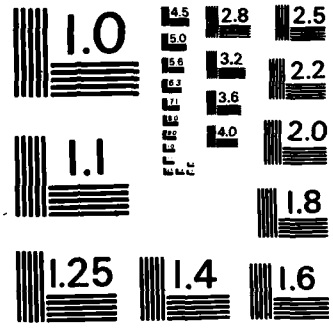
AD-A121 313 STORAGE REQUIREMENTS FOR FAIR SCHEDULING(U) YALE UNIV 1/1
NEW HAVEN CT DEPT OF COMPUTER SCIENCE
M J FISCHER ET AL. OCT 82 RR-251 N00014-82-K-0154
UNCLASSIFIED F/G 12/1 NL

END
FILMED
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

(12)

AD A121313

LUX ET VERITAS

This document has been approved for public release and sale; its distribution is unlimited.

DTIC FILE COPY

YALE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

82 11 09 094

STORAGE REQUIREMENTS FOR FAIR SCHEDULING

by

Michael J. Fischer and Michael S. Paterson

Research Report # 251

October, 1982

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: 251

2. GOVT ACCESSION NO.: A121 312

3. RECIPIENT'S CATALOG NUMBER:

4. TITLE (and Subtitle): STORAGE REQUIREMENTS FOR FAIR SCHEDULING

5. TYPE OF REPORT & PERIOD COVERED: Technical Report

6. PERFORMING ORG. REPORT NUMBER:

7. AUTHOR(s): Michael J. Fischer and Michael S. Paterson

8. CONTRACT OR GRANT NUMBER(s): ONR: N00014-82-K-0154 and NSF: MCS-8116678

9. PERFORMING ORGANIZATION NAME AND ADDRESS: Department of Computer Science/Yale University, Dunham Lab./10 Hillhouse Avenue, New Haven, Connecticut 06520

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: NR 049-456/11-5-81 410

11. CONTROLLING OFFICE NAME AND ADDRESS: NSF, Washington, D.C. 20550/ Office of Naval Research, 800 N. Quincy, Arlington, VA 22217

12. REPORT DATE: October, 1982

13. NUMBER OF PAGES: 7

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Office of Naval Research, 800 N. Quincy, Arlington, VA 22217, ATTN: Dr. R.B. Grafton

15. SECURITY CLASS. (of this report): Unclassified

15a. DECLASSIFICATION DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report): Approved for public release; distributed unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report):

18. SUPPLEMENTARY NOTES:

19. KEY WORDS (Continue on reverse side if necessary and identify by block number): Fair scheduling, analysis of algorithms, storage bounds, parallel computatio

20. ABSTRACT (Continue on reverse side if necessary and identify by block number):

A scheduler is strongly fair if each process which requests service infinitely often is served infinitely often, and it is weakly fair if each process which requests service all but finitely often is served infinitely often. We show that any strongly fair scheduling algorithm for n processes requires at least n! storage states (i.e. space proportional to n log n). Similarly, any weakly fair scheduling algorithm requires at least n storage states. Both bounds are optimal.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Storage Requirements for Fair Scheduling*

Michael J. Fischer
Yale University
New Haven, Connecticut

and

Michael S. Paterson
University of Warwick
Coventry, England

DTIC
COPY
INSPECTED
2

Manuscript Date
August 1982

Report Date
October 1982

A

Keywords: Fair scheduling, analysis of algorithms, storage bounds, parallel computation

*This work was supported in part by the Office of Naval Research under Contract N00014-82-K-0154, and by the National Science Foundation under Grant MCS-8116678.

1. Introduction

In [3], Park discusses notions of strong and weak fairness in the execution of guarded iterations. These concerns are also considered in [1] and [2]. We show that any "strongly fair" scheduling algorithm for n processes requires at least n! storage states (i.e. space proportional to n log n). Similarly, any "weakly fair" scheduling algorithm requires at least n storage states. Both bounds are optimal.

For our present purposes we may define a *scheduler* as a transducer A with an input alphabet of symbols corresponding to the non-empty subsets of {1, ..., n} and output alphabet {1, ..., n}. It has the property that for each symbol input the generated output symbol is an element of the corresponding subset. We may regard each input symbol as requests for service from some subset of n processes and the output given by A as the scheduler's choice of which one of these to *serve*. We consider infinite runs of such a scheduler.

A scheduler is

1. *strongly fair* if each process which requests service infinitely often is served infinitely often, and
2. *weakly fair* if each process which requests service all but finitely often is served infinitely often.

Thus at any time in a strongly (weakly) fair schedule any process will eventually be served if it requests service infinitely (continuously) from that time. Park's example of a strong scheduler in [P] keeps the processes in a queue. At each step it serves that requesting process which is earliest in the queue and then sends this process to the back of the queue. That this provides strongly fair scheduling is easy to see since when any process is unsuccessful in its request it advances one position in the queue. Park expresses disquiet at the implementation overheads for such a scheduler.

By contrast, he shows a simple economical weakly fair scheduler. A counter with values in {1, ..., n} is maintained. At each step the counter is incremented modulo n until it reaches the number of a process requesting service. This process is then served.

We shall show that both of the schedulers given by Park are optimal in their use of storage space.

2. Main Results

Theorem I. Any strongly fair scheduler for n processes has at least n! states.

Proof. For each i, let P_i be the set of scheduler states with the property that the next time process i requests service it will indeed be served.

Lemma 1. For $i \neq j$, $P_i \cap P_j = \phi$.

Proof. An immediate request for service by processes i and j would be an irreconcilable conflict for any state in $P_i \cap P_j$. □

Lemma 2. For all i, $P_i \neq \phi$.

Proof. Suppose $P_i = \phi$. Since the initial state is not in P_i, there is some sequence w_1 of inputs ending in a request from process i such that process i is not served during w_1. Since the resulting state is also not in P_i, the same reasoning produces a continuation w_2 with the same property. In this way we can show the existence of an infinite sequence of inputs $w_1 \cdot w_2 \cdot w_3 \cdot \ldots$ in which i requests service infinitely often but is never served. This contradicts strong fairness. □

Lemma 3. The set of states P_i is closed under the transitions effected by i-free inputs.

Proof. Immediate from the definition of P_i. □

The proof of Theorem I now proceeds by induction on n. The result is trivial for $n = 1$. Let us suppose the result holds for n-1 processes and consider the case of n processes.

With Lemma 2 in mind, consider any $s_i \in P_i$. With s_i as an initial state and allowing only i-free inputs, we find that we have a strongly fair scheduler for $\{1, \ldots, n\} - \{i\}$. This follows from the strong fairness of the original scheduler. By the inductive hypothesis this strongly fair $(n-1)$-scheduler uses at least $(n-1)!$ states, and by Lemma 3 all these states are in P_i. Hence $|P_i| \geq (n-1)!$. Since this inequality holds for each i, we have, using Lemma 1, that the original scheduler has at least n! states. □

Thus Park's strongly fair scheduler is optimal in its storage requirement. Indeed the naturalness of his queuing structure is supported by an analysis of the proof technique above. In a natural way we can associate with every permutation of the processes a disjoint non-empty subset of the scheduler states.

We close with a minor result, analogous to Theorem I.

Theorem II. Any weakly fair scheduler for n processes has at least n states.

Proof. Consider the (constant) input sequence in which each process requests service at every step. If the scheduler has fewer than n states, its resulting ultimately periodic behaviour has period less than n and so fails to serve some processor. □

Acknowledgement

We are grateful to David Park for introducing us to this problem.

References

[1] K. R. Apt and E.-R. Olderog. Proof rules dealing with fairness. Bericht Nr. 8104, Institut für Informatik u. Praktische Mathematik, Kiel University (1981).

[2] D. Lehmann, A. Pnueli, and J. Stavi. Impartiality, justice and fairness: the ethics of concurrent termination. In *Automata, Languages and Programming*, S. Even and O. Kariv, eds., Lecture Notes in Computer Science Vol. 115, Springer-Verlag, 1981, 264-277.

[3] D. Park. A predicate transformer for weak fair iteration. *Proc. Sixth IBM Symp. on Mathematical Foundactions of Computer Science*, IBM Japan (1981), 257-275.

DISTRIBUTION LIST

Office of Naval Research Contract N00014-82-K-0154
Michael J. Fischer, Principal Investigator

Defense Technical Information Center
Building 5, Cameron Station
Alexandria, VA 22314
(12 copies)

Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Dr. R.B. Grafton, Scientific
Officer (1 copy)
Information Systems Program (437)
(2 copies)
Code 200 (1 copy)
Code 455 (1 copy)
Code 458 (1 copy)

Office of Naval Research
Branch Office, Pasadena
1030 East Green Street
Pasadena, CA 91106
(1 copy)

Naval Research Laboratory
Technical Information Division
Code 2627
Washington, D.C. 20375
(6 copies)

Office of Naval Research
Resident Representative
715 Broadway, 5th floor
New York, N.Y. 10003
(1 copy)

Dr. A.L. Slafkosky
Scientific Advisor
Commandant of the Marine Corps
Code RD-1
Washington, D.C. 20380
(1 copy)

Naval Ocean Systems Center
Advanced Software Technology Division
Code 5200
San Diego, CA 92152
(1 copy)

Mr. E.H. Gleissner
Naval Ship Research and Development Center
Computation and Mathematics Department
Bethesda, MD 20084
(1 copy)

Captain Grace M. Hopper (008)
Naval Data Automation Command
Washington Navy Yard
Building 166
Washington, D.C. 20374
(1 copy)

Defense Advance Research Projects Agency
ATTN: Program Management/MIS
1400 Wilson Boulevard
Arlington, VA 22209
(3 copies)

Professor Michael S. Paterson
Department of Computer Science
University of Warwick
Coventry, Warwickshire CV4 7 AL
England
(1 copy)