_HSC50$DUA11:[GOLDEN.BELLENOT.CUBE.OBJ]AAREADME.UPDATE;2

JET PROPULSION LABORATORY

INTEROFFICE MEMORANDUM
SFB:363-86-002

Update notes for the dining philosophers

12-aug-86

steve bellenot

the philosophers are pretty much in the state they were last summer with exception of many compile time switches. (hence the number of different c files.)

the one error noticed is that each philosopher has one more thought than MAX_NUM_THOUGHTS (perhaps it is a metathought). one should be careful to note this #def constant. some phil's have this set to 1, others 5 and a third has it at 99. (those with 99 thoughts are in ltphil.c and sltphil.c)

another switch in the glutton #def, if on (as is currently true) the philosophers spend of their time hungry and hence waiting for forks. if glutton isn't #def they spend most of their time thinking and the forks are mostly free.

a query switch is available as well. a qphil.c will query one of the forks before and after it requests that fork.

forks and phils _pprintf every time there event section is run. the sforks and sltphils are silent versions of the same objects.

an interesting .cf file is one with one object of type phil with name FOO and initial message AFOOFOO\0 will happily run by itself. using qphil or ltphil etc in place of phil will also work.

the new object added this summer is the timekeeper object and its many variants. the timekeeper is a "prime mover" object which also "almost" times the simulation. its design as a timekeeper is wrong but it will give a rough idea of the simulation time. The prime mover part was required for timing with 2113, and may not be needed with versions 2214 and above.

in any case time4a.c is a timekeeper which starts 4 phils named phil0, phil1, phil2, and phil3 each of which is told to delay 'A' amount (which is zero). on the ih this file is time4A.img since a delay of 'a' (around 30) is also possible. how these delays work is in my inter office memo 363:85:002 the dining philosopher benchmark. (from last summer)

JET PROPULSION LABORATORY INTEROFFICE MEMORANDUM
SFB:363-86-003

TO:	Time Warp List	15 September 1986
FROM:	Steve Bellenot	
SUBJECT:	Timing Philosophers on Hypercubes	

This memo contains the "best" hypercube timing measurements on the dining philosopher benchmark that I obtained this last summer. These results are preliminary. The reader is warned to read all the cautions. Some supporting VAX timing is also included.

These measurements seem to indicate that efficiencies near 90% and speedups of around 3.6 were obtained by comparing the 4-node to 16-node runs (runs C to B or D to E, see [9] below). There is a timing which indicates the cost of communicating over many hops (compare run F to B, see [11]). And finally, a couple of timings that imply that a goodly percent of the time was spend in operating system overhead (compare run E to B or run D to C and the VAX timings below [12]). However there are many cautions on why and how these timings are limited (i.e. why comparing run A to B gives an efficiency of over 100% [10]).

TABLE 1: The Hypercube Timing Results

RUN	CLOCK	TIMEKEEPER*	GVTTIME*	DATE	CONFIG	DELAY
A	41 min	1894.95	1901.71	11 Aug 86	tnice2.16	0
B	10 min	466.37	478.60	11 Aug 86	tnice4.16	0
C	9 min	411.20	425.25	12 Aug 86	toth2.4	0
D	9 min	435.02	448.63	12 Aug 86	toth2.4	8
E	10 min	503.07	508.07	12 Aug 86	tnice4.16	8
F	13 min	553.40	569.36	13 Aug 86	tfar4.16	0

*Time in seconds measured by a "slow" clock

TABLE 2: The VAX Timing Results

DELAY	#THOUGHTS	ISE860*	ISE780*	ISE752*
0	5	0.802	2.767	4.951
8	5	1.575	5.829	9.525
0	10	1.017	3.713	6.797
8	10	2.420	9.333	15.106

*Time in seconds

EXPLANATIONS, ASSUMPTIONS, CONCLUSIONS AND ITEMS FOR YOUR CONSIDERATION:

1. The slow clock in Table 1 was assumed to be uniformly slow. Both the timekeeper and gvttime are based on this slow clock. The timekeeper's time is the time from before he sends all the initial messages to each of the philosophers to the time after he receives the last message from each philosopher. (Actually the timekeeper's time may not be the total time of the simulation. If some philosopher "sends" his final message twice, Time Warp wouldn't send the second message.) Timekeeper's time is a lower bound on the total simulation time. Gvttime is the time between the gvtinit message and the time when gvt first becomes positive infinity. Gvttime is an upper bound on the total simulation time. The time between gvt calculations is on the order of ten seconds. Thus even though the slow clock is "accurate to 100 microseconds" these times in Table 1 are perhaps accurate to only a couple of significant figures.

2. The clock time in Table 1 was measured by an old digitial wristwatch. This watch didn't have a stopwatch feature, and so times are rounded off to the nearest minute. This clock was attempting to measure the same time interval as the timekeeper object. These two measurements imply that the internal Time Warp clock was warped when these runs were made. However all these runs were made after Mike and I found the big bug with the timers and the timer code didn't change over these runs.

3. The times in Table 2 were obtained by averaging the times of ten runs on each of the three VAX's ise860, ise780 and ise752. The runs were all within 6.5% of the given averages. The initial measurements were said to be accurate to 10 milliseconds. These timings of the dining philosopher's benchmark indicate that a delay of 8 roughly doubles the total time of running the simulation on the Time Warp Simulator. All these VAX timings were run at night.

4. The dates are given in Table 1 partly because the Time Warp code was changing over this time period. Indeed the run F was attempted on 11 August but it wouldn't complete. The "fix" in forwarding messages wasn't in the code until the 13th. However, runs A-E don't really use the message forwarding code since their communication is between nearest neighbors only. We assume these code improvements did not significantly change the total simulation time.

5. These timings in Table 1 are not averages. Each timing was made only once. (On a time available basis between time spent debugging the code.) However, earlier experience with multiple timings of small simulations using Time Warp on the hypercube had shown no significant variation on total simulation time.

6 We assume the changes in the parameters given here do not effect our relative timing conclusions. Maximum_num_thoughts was 100 for each of the hypercube runs and this parameter is given by #thoughts in Table 2. This #defined constant is one more than the number times the philosopher loops throught the think-eat cycle. Also there were only five philosophers and five forks in the VAX runs but either four philosophers and four forks or sixteen philosophers and sixteen forks on the hypercube runs. We state these differences in case anyone is tempted to compare VAX times with hypercube times on an absolute basis.

7. The configuration files names are given in Table 1 were on the IH nncp1 when i left this summer (in /usr/steve/mon_dine). Let me summarize their contents. First each filename ends with two numbers x.y, where x is the dimension of the hypercube and y is the number of philosopher objects and y is also the number of fork objects. Each philosopher communicates to only two forks, which we will call his left and right forks.

tnice4.16: One philosopher per node, his left fork is on the same node and his right fork is one hop away on a nearest neighbor node. Sixteen philosophers, sixteen forks and sixteen nodes.

tnice2.16: Four philosophers per node, each philosopher has his left fork on the same node and his right fork is one hop away on a nearest neighbor node. Sixteen philosophers, sixteen forks and four nodes.

toth2.4: One philosopher per node, his left fork is on the same node and his right fork is one hop away on a nearest neighbor node. Four philosophers, four forks and four nodes.

tfar4.16: One philosopher per node (also one fork per node), his left fork is as far away as possible, four hops away, and his right fork is three hops away. (Thus communication is as bad as possible for the dining philosophers.) Sixteen philosophers, sixteen forks and sixteen nodes.

8. We assume the simulation took long enough (or the time measurements are coarse enough) that the initialization (and finalization) "objects" (timekeeper and stdout) are not important factors in these timings. (Each configuration file included these two objects as well.)

9. We assume that a reasonable speedup time is obtained by taking the total time to run toth2.4 dividing by the total time to run tnice4.16 and multipling the result by four. (Somehow it seems reasonable to say there is "exactly" four times as much work in tnice4.16 as there is in toth2.4.) Note that there are reasons why one could think otherwise. For example, the world map is four times larger with four times the objects. Seaching time for world map records is roughly the same, but more memory is being used to store them.

10. We believe the reason why the speedup time obtained by dividing run A's time by run B seems larger than possible is due to memory exhaustion. In run A, with four times as many objects on each node, each processor spend a larger percentage of its time between gvt calculations out of memory, than did each processor in run B. (Objects always tried to save their state after each event and the states were much bigger than messages in these versions of the code.) Also changing the time interval between gvt calculations might have improved things, however that was not attempted.

11. Note that the one-third increase in total time between runs B and F could be entirely due to increased communication time, or due to an increased number of rollbacks because of increasing asynchronous delays. A bit of statistics would likely determine the combination of effects that were timed. In any case we have at least an example of where incorrect placement of objects results in a 30% increased in execution time, even when both configurations are load balanced.

12. Perhaps the most depressing timings are C vs D and B vs E. The VAX timings would say we have doubled the amount of the time needed to complete the simulation, yet the increased time is less than 10%. Does this imply that 90% of the time in runs B and C is overhead needed to go from the sequential simulator to the parallel Time Warp operating system? Of course, it is way to early to be depressed by such results since the main attempt at optimalization is (was?) yet to be done when these timings were obtained.