John H. Kooi
999-99-9999
CS546-92
Location UP72
10/5/2000

Project Proposal
Parallelizing Chemical Engineering Simulations with MPICH
(Pressure Swing Adsorption)

Purpose:

The execution of this project serves an extended purpose. My company, UOP LLC, designs processes for the Oil Refining and Petrochemical Industries. One of these designs is called Pressure Swing Adsorption (PSA).

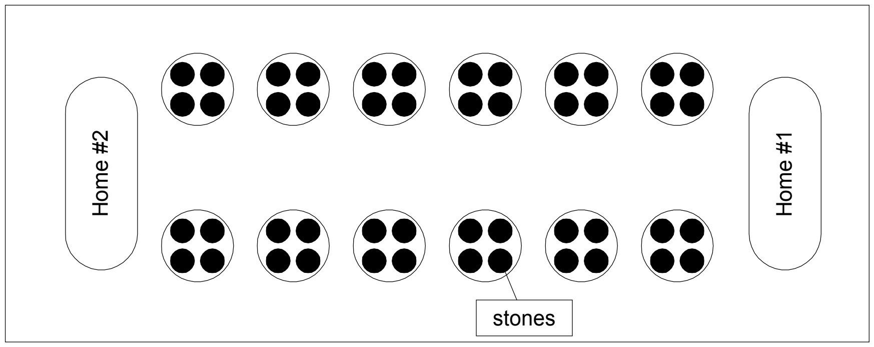
In a PSA process, a gas stream containing a mixture of different gases is separated or purified. In particular, very pure hydrogen can be obtained by removing impurities such as methane, carbon dioxide, etc. from a mixture of gases containing mostly hydrogen. This separation is accomplished by passing the gas stream through a vessel containing an absorbing material (like silica gel or molecular sieves). The chemical basis of this separation is based on the fact that the species present in the gas are absorbed at different rates and that these rates are affected by the *pressure* of the system. By clever adjustment of the pressure (swinging the pressure), impurities are first absorbed and then later flushed from the absorbent.

In order to design a PSA process, chemical engineers at UOP run two different programs. One, called Multicomponent Absorption Simulation Calculation (MASC), runs relatively quickly, on the order of 10 minutes. MASC performs some rule-of-thumb calculations, selected to provide satisfactory answers in a reasonable amount of time. The second program, called Polybed Adsorption System Simulation (PASS), performs a rigorous set of calculations, providing more accurate answers, but in a less reasonable amount of time, approximately 2 hours.

It is the second program, PASS, that I would like to parallelize in order to reduce its execution time. However, this effort will require on the order of 6 to 12 months of full-time work, so I propose to parallelize a simpler program in order to learn the concepts and practices that I will need to apply to PASS.

Strategy:

And this is the simpler program: in the game of Mancala, a wooden board is arranged so that each of the two players has a “home” at the right end of the board. In addition, each player begins her turn by picking up all of the stones from one of six holes on her side of the board. Then the player drops the stones, one-by-one, in the holes in a counter-clockwise fashion, making sure to drop one in her “home” but not in her opponent’s.

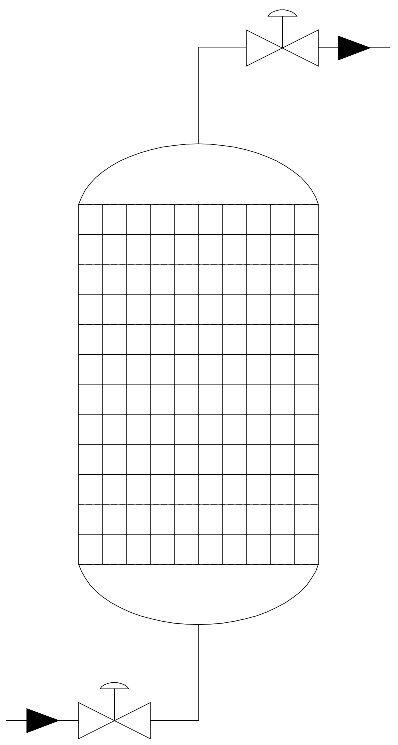


The opportunity for parallelization is obvious. Since the player can choose from six different holes in order to begin play, in a recursive parallelism model, this decision could easily be divided among six processors.

Two more rules make the problem interesting. If the active player drops her last stone in a hole that is not empty, she then picks up all of the stones from that hole and continues. If the active player drops her last stone in "home", she continues play by again selecting all of the stones from one of the 6 holes on her side of the board.

Of course at any point during play, one or more of the six holes may be empty. This presents an interesting problem: during execution, a varying number of processors may be required in order to decompose the problem as it is solved. In all likelihood, the number of processors will be less than the number of threads possible during execution.

It is this feature which would find direct application to the design of a PSA process using the PASS program. In rigorous calculations, a PSA vessel containing absorbent is divided into a grid:



This grid is created to model the flow of the gases up through the absorber (thus, the horizontal layers) and also to model the axial dispersion of the gases in the absorbent (thus, the vertical layers). Although the granularity of the grid determines the accuracy of the model, the granularity is chosen by the user at execution time. This means that a parallelization of the simulation calculations must adapt itself to the number of processors available. In all likelihood, the number of processors available will be less than the granularity of the grid. This is the similarity to the Mancala model which I hope to employ.

In addition, outside of class, I would like to eventually investigate the form of clustering available by using an existing local area network of desktop computers which can be used collectively when the computers are not serving their respective users. Known either as “cycle harvesting” or “workstation farms”, this form of cluster can provide a valuable resource under certain circumstances. I feel this is possible at UOP because, as I look at my own Windows NT workstation this morning, the “System Idle Process” has logged over 125 hours, “Findfast.exe” has logged 10½ hours, and Microsoft Word is in third place with only 31 minutes of CPU time. Figures like these suggest that “cycle harvesting” is feasible.

<u>Reference</u>:

[1] Gas Separation by Adsorption Processes, Ralph T. Yang, Imperial College Press, 1999
[2] Principles of Adsorption and Adsorption Processes, Douglas M. Ruthven, John Wiley & Sons, 1984
[3] MPICH Model MPI Implementation Reference Manual (Draft), William Gropp, Nathan Doss, December 2, 1999. Available from http://www-unix.mcs.anl.gov/mpi.
[4] MP-MPICH, User Documentation & Technical Notes (Draft), Joachim Worringen, Karsten Scholtyssik, RWTH Aachen. Lehrstuhl fur Betriebssysteme, 6/14/00. Available from http://www.lfbs.rwth-aachen.de/~joachim/MP-MPICH.html.
[5] How to Build a Beowulf, A Guide to the Implementation and Application of PC Clusters, Thomas L. Sterling, John Salmon, Donald J. Becker, Daniel F. Savarese, The MIT Press, 1999
[6] Parallel Programming with MPI, Peter S. Pacheco, Morgan Kaufmann Publishers, Inc., 1997.