Computers and Chemical Engineering xxx (2017) xxx–xxx

Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

Parallelization methods for efficient simulation of high dimensional population balance models of granulation

Franklin E. Bettencourt, Anik Chaturbedi, Rohit Ramachandran*

Department of Chemical & Biochemical Engineering, Rutgers, The State University of New Jersey, 08854 Piscataway, NJ, USA

ARTICLE INFO

Article history:
Received 3 October 2016
Received in revised form 20 February 2017
Accepted 22 February 2017
Available online xxx

Keywords:
MPI
OpenMP
Parallel computing
Population balance model
Granulation
Pharmaceutical process design

ABSTRACT

In order to solve high resolution PBMs to simulate real systems, with high accuracy and speed, a comprehensive and robust parallelization framework is needed. In this work, parallelization using just Message Passing Interface (MPI) and a more advanced method using a hybrid MPI + OpenMP (Open Multi-Processing) technique, have been applied to simulate high resolution PBMs on the computing clusters, SOEHPC and Stampede. We study the speed up and the scale up of these parallelization techniques for different system sizes and different computer architectures to come up with one of the fastest ways to solve a PBM to date. Parallel PBMs ran approximately 50–60 times faster, when using 128 cores, than the serial PBMs ran. In this work it is found that hybrid MPI + OMP methods which account for socket affinities led to the fastest PBM compute times and about 80% less memory than a purely MPI approach.

1. Introduction and objectives

Particulate processes represent half of all industrial chemical production (Seville et al., 1997). Some examples of particulate processes include crystallization (Sen et al., 2014), granulation (Barrasso et al., 2015), milling (Barrasso et al., 2013a) and biological processes (Ramakrishna and Singh, 2014). Some important products made using particulate processes are fertilizers, detergents, aerosols, and pharmaceuticals (Prakash et al., 2013b). Due to the prevalence of particulate processes there is a great demand for methods, which can be used to aid in the detailed study and design of these processes.

The accurate modeling of particulate systems is crucial to enhance process development, production, equipment design, optimization, and process control (Ramachandran and Barton, 2010). However, these systems are very difficult to model due to complex micro phenomena that take place within them (Muzzio et al., 2002). Population balance models (PBMs) have proven to be one of the most efficient ways to model these systems while still capturing much of the physics that takes place in them (Ramakrishna and Singh, 2014).

Even though PBMs are one of the most efficient (fastest) ways to model these processes, high resolution and high dimensional models can still take days to solve and optimize via the use of conventional desktop/workstations (Prakash et al., 2013b). High resolution refers to the classification of particles into a large number of size classes (i.e. bins) and the division of the granulator volume into fine segments and high dimensional implies multiple internal/external variables of high resolution grids. To optimize the design of a process, or to tune a model based on a real system, a PBM will need to be run tens or hundreds of times creating a great need for faster simulations. There is also the push to increase PBM speeds to utilize them for model predictive control (MPC) and feed forward controllers (Christofides et al., 2007). In the past when scientists have faced computationally intensive problems they would use many CPUs working together to solve the problem faster, a method referred to as parallel programming (Wilkinson and Allen, 1997). Using modern computing clusters to evaluate a PBM in parallel has a great deal of promise. Two applications used for parallel programming are Message Passing Interface (MPI) and Open Multi-processing (OMP) both of which have different strengths and weaknesses.

1.1. Objectives

The overall objective of this study is to develop more efficient parallelization methods to reduce the computational times of high-dimensional PBMs. To that effect, specific objectives are listed below:

* Corresponding author at: 98 Brett Rd, Piscataway Township, NJ 08854, USA.
E-mail address: rohitrr@rci.rutgers.edu (R. Ramachandran).

http://dx.doi.org/10.1016/j.compchemeng.2017.02.043
0098-1354/© 2017 Elsevier Ltd. All rights reserved.

- MPI and hybrid MPI + OMP methods were used to parallelize a PBM.
- The performance of the parallelized models was tested using different criteria such as speed up, parallel efficiency, and memory usage.
- The hybrid MPI + OMP approach was compared to the purely MPI approach and was also compared with other works in the literature.

2. Background and motivation

2.1. Population balance models

PBMs have been effectively used to study granulation with a great deal of accuracy. Granulation is a process for engineering particles via liquid or solid binders to form larger aggregate granules with desired traits such as particle size distribution (PSD) and bulk density. The mechanism of particle growth and breakage are referred to as rate processes (Barrasso et al., 2013b). From a process stand point PBMs are used to simulate how the distribution of a set of particles, with varying properties, will change due to the systems rate processes over time (Barrasso et al., 2013b). However more generally, PBMs evaluate how a population distribution of entities is effected by its environment over time (Ramakrishna and Singh, 2014). A general form of population balance model, which applies to granulation systems, is shown as Eq. (1) below (Barrasso et al., 2013b),

$$\frac{\partial F(x,z,t)}{\partial t} + \frac{\partial}{\partial x}\left[F(x,z,t)\frac{dx}{dt}(x,z,t)\right] + \frac{\partial}{\partial z}\left[F(x,z,t)\frac{dz}{dt}(x,z,t)\right] = \Re_{formation}(x,z,t) - \Re_{depletion}(x,z,t) + \dot{F}_{in}(x,z,t) - \dot{F}_{out}(x,z,t) \quad (1)$$

where x is a vector which represents some or all of the characteristic properties of the particles in the system. Here x is a vector of internal parameters and is used to represent solid (s), liquid (l), and gas (g) contents of the particles. Here, z is a vector of external coordinates and represents spatial variance in the system of interest. F represents the population densities of particles described by the vectors x and z.

The first term on the left accounts for the rate of change of the particle number density, with time. The second term from the left, represents the rate of changes of population densities as the values of x are changed due to growth terms (dx/dt). In granulation systems this term would be associated with layering, consolidation, and liquid addition. The third term describes the change in population density over the physical space of the system due to movement, dz/dt is the particle velocity. For a continuous system F_{in} and F_{out} are the rates of particle inflow and outflow respectively. For a batch process F_{in} and F_{out} are both set to zero. The $\Re_{depletion}$ and $\Re_{formation}$ describe the net changes due to the rate processes of nucleation, aggregation, and breakage (Barrasso et al., 2013b).

2.2. Parallelization and parallel computing

2.2.1. Overview

In general parallel computing is taking a problem, breaking it into parts which can be solved independently, and distributing those small independent parts to many computers/computing cores to be evaluated all at the same time, i.e. in parallel (Wilkinson and Allen, 1997).

2.2.2. Computer architecture

Computing clusters are made up of many nodes connected by means of some sort of communication network such as InfiniBand, a type of high speed wireless communication, or Ethernet. Each node is analogous to a PC (personal computer) in that it has one or more CPUs (central processing unit), RAM (random access memory), a motherboard, cooling systems, and possibly GPUs (graphics processing unit) or co-processors. Commonly CPUs are multi-core processors, which means they have multiple compute units or cores, which can carry out independent computations. CPUs come with their own built-in memory which is very fast and heavily used for computation which is called the cache. Another important type of memory is the RAM, CPUs will have a direct connection through the motherboard's CPU socket to the RAM. RAM is slower than cache memory and therefore efficient memory usage, which optimizes cache utilization and minimizes data transfers from the RAM to the CPU and from the CPU to the RAM is critical for fast computation. Large amounts of memory transfer from the RAM to the CPU can greatly limit the speed at which computation can be performed. Furthermore, data movement in general is one of the greatest limitations to the performance of a parallel application. For the previous reasons to maximize performance network message passing should be minimized and when memory is needed cache utilization should be optimized.

Computer architectures, are often classified as distributed memory machines, shared memory machines, or some sort of combination of distributed and shared. Clusters are actually distributed memory systems (nodes have separate memory from one another) with shared memory subsystems (the CPU cores on a node can operate in shared memory mode). Additional architectural features arise when a node has more than one CPU, most nodes these days support two CPUs. Even though a node can be operated in shared memory, the CPUs on the node cannot access all of the memory on that node at the same speeds. One CPU can access memory that it is connected to through its socket much faster than it can access memory stored in a memory bank associated with the other CPU socket on the node. These kinds of systems are called non-uniform memory access (NUMA) clusters (Jin et al., 2011). The compute cores of a single CPU can all potentially access the same RAM at the same speed since they are all connected to it through the same socket. Since the cores on a CPU can all draw from the same RAM they share the memory, and are referred to as a shared memory system. Two nodes are distributed machines with respect to another because the cores on one node do not share the same physical RAM as the cores on the other node. Due to this the interplay between computer architecture, software, and computational load distribution need to be considered when developing a parallel program (Adhianto and Chapman, 2007).

2.2.3. Parallelization applications and practices

Message passing interface (MPI) and open source message passing (OMP) are two application program interfaces, which have been commonly used to parallelize computations. MPI is an application which is used for distributed memory computing (Jin et al., 2011). When just MPI is used, all of the CPU cores on all of the nodes have their own private local memory, meaning core-to-core communication must be done by explicit message passing even on hardware that can utilize shared memory (Jin et al., 2011). Large amounts of message passing can lead to bottlenecking due to message passing overhead and due to overloading of the bandwidth/communication network (Jin et al., 2011). Using only MPI results in each core needing its own copy of all variables used for the computation it is performing, resulting in inefficient memory usage due to large numbers of duplicate variable storage. Even on shared memory architecture systems MPI will operate each core as a distinct unit with its own private memory.