74.785 Paper Presentation

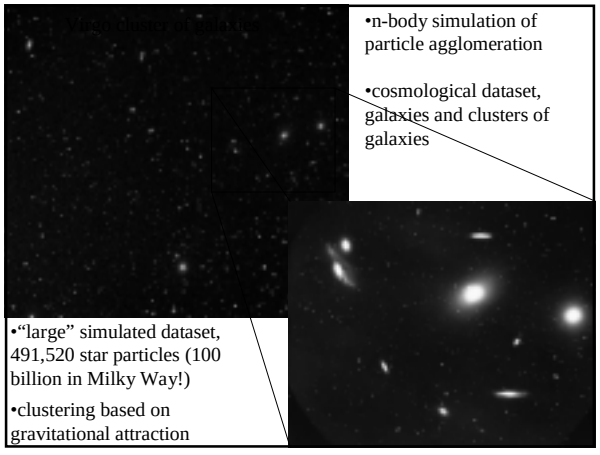
"Design and Evaluation of a Parallel HOP Clustering Algorithm for Cosmological Simulation"

Ying Liu, Wei-keng Liao, Alok Choudhary
Department of Electrical and Computer Engineering
Northwestern University

Summary and Presentation
by
Rodrigo Vivanco

Outline

- Problem to be solved
  - N-body simulation
- Proposed solution
  - HOP clustering using KD trees
- Parallelization of algorithm
  - Domain partition and MPI
- Results
  - IBM SP2, SGI Origin 2000, Linux cluster
- Discussion



•n-body simulation of particle agglomeration

•cosmological dataset, galaxies and clusters of galaxies

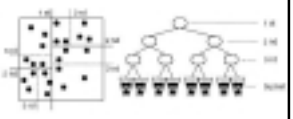
•"large" simulated dataset, 491,520 star particles (100 billion in Milky Way!)

•clustering based on gravitational attraction

Solution

- HOP clustering algorithm
  - assigns 3D coordinates and a density to each particle
  - search within spatial neighbourhood for neighbour with higher density
  - through neighbour can "hop" to another higher density particle
  - a cluster consist of all particles that hop to the same highest density local maxima

HOP clustering



- Constructing kd-tree structure
  - balanced tree, leaf nodes (buckets) contain particle data, less particles in buckets, greater the tree depth
  - recursively partition particles in 3D space
  - particles close in 3D space in same bucket or sibling buckets -> efficient spatial neighbourhood search
- Generating density estimates
  - search kd-tree for N densest neighbours most, use SMOOTH to calculate the mean density, computationally intensive part of HOP algorithm

HOP clustering cont…

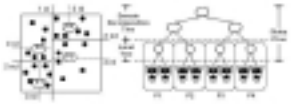
- Hopping
  - after particle density calculated, associate with most dense neighbour (num neighbours user provided)
  - once at denser neighbour, search again for a more dense neighbour
  - stop when no neighbour with higher density
- Grouping
  - particles that terminate at the same dense neighbour, belong to the same cluster
  - groups may be merged or disbanded

HOP parallelization

- most expensive operation, searching spatial neighbourhood for dense neighbours and calculating neighbourhood density
- Use domain partition, kd-tree buckets assigned to different processors
- balance computational load by assigning to each processor N/P particles
- kd-tree property that particles near in space are in the same bucket or sibling buckets increases locality and reduces communication costs

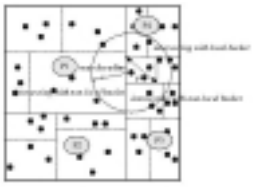
HOP parallelization cont…

- Constructing kd-tree
  - partition sub-domains until equals number of processors



  - each processor build its own local kd-tree, no inter-process communication needed at this step
  - after partition, broadcast local tree to all processors, copy of global tree to reduce communication for testing if particle neighbourhood crosses partition boundaries
  - after kd-tree construction, particles distributed based on spatial locality, and load will be balanced

HOP parallelization cont…



- Density estimates
  - based on N-nearest neighbours
  - communication is required if neighbour owned by another processor
  - one message needed, get information for all neighbour particles in a different processor at once
  - in a kd-tree, neighbours will be “close”, except for highly irregular distributions

HOP parallelization cont…

- Hopping
  - keep track of all the particles that “hopped” to remote processor, one communication request
- Grouping
  - each processor builds boundaries based on local particles, local boundaries broadcast to others
  - Particles with density below threshold excluded, groups with boundaries within threshold merged.

Software development

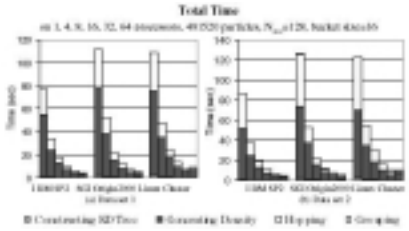
- C programming language with MPI
- Used the Single Program Multiple Data (SPMD) model
- IBM SP2 (San Diego Supercomputing Center)
  - from 2 to 512 POWER2 Architecture RISC System/6000 processor nodes, each with its own private memory, interconnected by a switched network
  - multi processing is based on shared-distributed memory and message-passing
  - famous SP2 is Deep Blue
- SGI Origin 2000 (National Center for Supercomputing Application)
  - Scalable Distributed Shared Memory Multiprocessor
- Linux Cluster (Argonne National Lab)

Results

- Varied the following parameters that affect 90% of clustering performance:
  - bucket size; larger buckets, more local particles
  - number of neighbours; larger N, more computation and communication
- Varied processors from 1 to 64 to evaluate scalability
- Two datasets that simulate galaxy formation
  - early in evolution, homogeneous distribution, later in evolution, more irregular distribution of particles

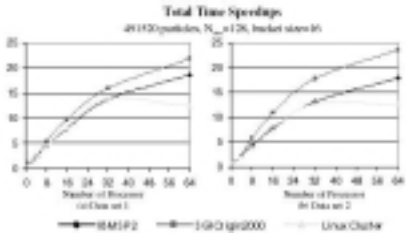
Results cont…

- Overall performance scales well on IBM SP2 and Origin 2000
- As expected, irregular particle distribution (2nd dataset) takes longer to cluster due to more inter-process communication

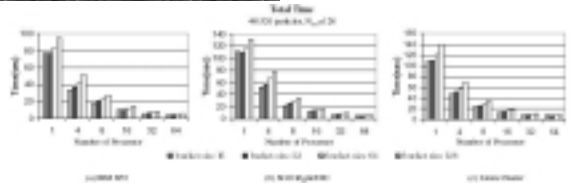


Results cont…

- Speed up degrades after 32 processors on Linux cluster; more processors => less local particles, decreases computation & increases communication

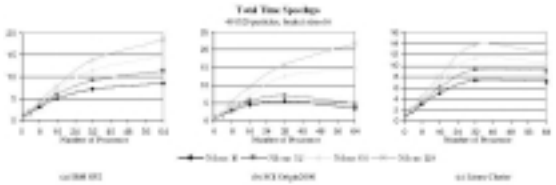


Results cont…



- Increasing bucket size
  - Larger bucket size, shorter time to construct kd-tree
  - Increases communication for density calculation and hopping
  - Execution time increases with larger bucker size

Results cont…



- Increasing number of density neighbours
  - **Execution** time increases as density calculation increases
  - **Speed up** increases, communication costs does not increase as much as calculation costs
- 24x speed up using 64 processors

Discussion

- Domain partitioning of particles across processors
- Uses kd-tree for fast neighbourhood search
- Uses inter-process communication to get “remote” particles before calculating densities and hopping
- Computation bottleneck is density calculations
- Increasing neighbourhood results in better speed-up, 24x with 64 processors

References

- D.W. Pfitzner, J.K. Salmon, T. Sterling, “Halo World: Tools for Parallel Cluster Finding in Astrophysical N-body simulations,” *Data Mining and Knowledge Discovery*, 419-438, Vol. 1, No. 4, 1997
- D.J. Eisentein, P. Hut, “HOP: A New Group Finding Algorithm for N-Body Simulation,” *Astrophysics.* J. 498, 137-142, 1998.
- J.L. Bently, “Multidimensional Binary Search Trees Used for Associative Searching,” *Communication of the ACM*, 18(9), September 1975.