Advanced Topics in Parallel Molecular Dynamics

Aiichiro Nakano

Collaboratory for Advanced Computing & Simulations
Department of Computer Science
Department of Physics & Astronomy
Department of Chemical Engineering & Materials Science
University of Southern California

Email: anakano@usc.edu
Load Balancing

Aiichiro Nakano

Collaboratory for Advanced Computing & Simulations
Department of Computer Science
Department of Physics & Astronomy
Department of Chemical Engineering & Materials Science
University of Southern California

Email: anakano@usc.edu
Load Balancing

- **Goal:** Keep all processors equally busy while minimizing interprocessor communication for irregular parallel computations

- **Issues:**
  - Spatial data vs. generic graph
  - Static vs. adaptive
  - Incremental vs. non-incremental

- **Load-balancing schemes:**
  - Recursive bisection
  - Spectral method
  - Spacefilling curve
  - Curved space
  - Load diffusion
Data Locality in Parallelization

Challenge: Load balancing for irregular data structures

Irregular data-structures/processor-speed

Map

Parallel computer

Optimization problem:
- Minimize the load-imbalance cost
- Minimize the communication cost
- Topology-preserving spatial decomposition
  → structured 6-step message passing minimizes latency

\[
E = t_{\text{comp}} \left( \max_p \left| \{ i \mid r_i \in p \} \right| \right) + t_{\text{comm}} \left( \max_p \left| \{ i \mid \|r_i - \partial p\| < r_c \} \right| \right) + t_{\text{latency}} \left( \max_p \left[ N_{\text{message}}(p) \right] \right)
\]
Computational-Space Decomposition

Topology-preserving “computational-space” decomposition in curved space

Curvilinear coordinate transformation

\[ \xi = x + u(x) \]

Particle-processor mapping: regular 3D mesh topology

\[
\begin{align*}
 p(\xi_i) &= p_x(\xi_{ix})P_yP_z + p_y(\xi_{iy})P_z + p_z(\xi_{iz}) \\
p_\alpha(\xi_{i\alpha}) &= \left[\xi_{i\alpha}P_\alpha / L_\alpha\right] \quad (\alpha = x, y, z)
\end{align*}
\]

Wavelet-based Adaptive Load Balancing

- Simulated annealing to minimize the load-imbalance & communication costs, $E[\xi(x)]$
- Wavelet representation speeds up the optimization

$$\xi(x) = x + \sum_{l,m} d_{lm}\psi_{lm}(x)$$

A. Nakano, *Concurrency: Practice and Experience* 11, 343 (’99)
Load Balancing as Graph Partitioning

- **Need:** Decompose tasks without spatial indices

- **Graph partitioning:** Given a graph $G = (N, E, W_N, W_E)$
  - $N$: node set = $\{j \mid \text{tasks}\}$
  - $W_N$: node weights = $\{w_N(j): \text{task costs}\}$
  - $E$: edge set = $\{(j,k) \mid \text{messages from } j \text{ to } k\}$
  - $W_E$: edge weights = $\{w_E(j,k): \text{message sizes}\}$

  choose a partition $N = N_1 \cup N_2 \cup \ldots \cup N_P$ to minimize
  - $\max_p\{\sum_{j \in N_p} w_N(j)\}$
  - $\max_{(p,q)}\{\sum_{j \in N_p, k \in N_q} w_E(j,k)\}$

- **Graph bisection:** Special case of $N = N_1 \cup N_2$

- **Choosing optimal partitioning is known to be NP-complete → need heuristics**

Prof. James Demmel (UC Berkeley)
Spectral Bisection: Motivation

1. Graph as point masses connected via harmonic springs
2. The node of the eigenvector of the Hessian matrix, $\partial^2 V/\partial x^2$, corresponding to the 2nd smallest eigenvalue separates the graph into 2
Spectral Bisection

Laplacian matrix:

$L(G)$ of a graph $G(N,E)$ is an $|N|$ by $|N|$ symmetric matrix:

- $L(G)(i,i) =$ degree of node $i$ (number of incident edges)
- $L(G)(i,j) = -1$ if $i \neq j$ and there is an edge $(i,j)$
- $L(G)(i,j) = 0$ otherwise

Theorems:

1. The eigenvalues of $L(G)$ are nonnegative:
   $\lambda_1 = 0 \leq \lambda_2 \leq \cdots \leq \lambda_N$
2. $\lambda_2(L(G)) \neq 0$ if and only if $G$ is connected

Spectral bisection algorithm:

1. Compute eigenvector $v_2$ corresponding to $\lambda_2(L(G))$
2. For each node $i$ of $G$
   a. if $v_2(i) < 0$, put node $i$ in partition $N$-
   b. else put node $i$ in partition $N+$

Example:

\[
\begin{bmatrix}
1 & 1 & -1 \\
-1 & 2 & -1 \\
-1 & 2 & -1 \\
-1 & 2 & -1 \\
-1 & 1 & 1
\end{bmatrix}
\]
$O(N) \lambda_2$ Computation

Lanczos algorithm:

- Given an $N \times N$ symmetric matrix $A$ (e.g., $L(G)$), compute a $K \times K$ “approximation” $T$ by performing $K$ matrix-vector products, where $K \ll N$
- Approximate $A$’s eigenvalues & eigenvectors using $T$’s

Choose an arbitrary starting vector $r$

- $b(0) = ||r||$
- $j=0$
- repeat
  - $j=j+1$
  - $q(j) = r/b(j-1)$
  - $r = A*q(j)$
  - $r = r - b(j-1)*v(j-1)$
  - $a(j) = v(j)^T * r$
  - $r = r - a(j)*v(j)$
  - $b(j) = ||r||$
- until convergence

$$T = \begin{bmatrix}
a_1 & b_1 \\
b_1 & a_2 & b_2 \\
& \ddots & \ddots & \ddots \\
& b_{K-2} & a_{K-1} & b_{K-1} \\
& & b_{K-1} & a_K
\end{bmatrix}$$
Multilevel Partitioning

Recursively apply:

1. Replace $G(N,E)$ by a coarse approximation $G_c(N_c,E_c)$, & partition $G_c$
2. Use partition of $G_c$ to obtain a rough partitioning of $G$, then uncoarsen & iteratively improve it

$\text{(N+}_c,\text{N-}_c) = \text{Multilevel\_Partition}(N_c,E_c)$
$\text{// returns N+ and N- where N = N+ } \cup \text{ N-}$
$\text{if } |N| \text{ is small}$
$1\text{ Partition G = (N,E) directly to get N = N+ } \cup \text{ N-}$
$\text{Return (N+,N-)}$
$\text{else}$
$2\text{ Coarsen G to get an approximation G}_c = (N_c,E_c)$
$3\text{ (N}_c+,\text{N}_c-) = \text{Multilevel\_Partition}(N_c,E_c)$
$4\text{ Expand (N}_c+,\text{N}_c-) \text{ to a partition (N+}_c,\text{N-}_c) \text{ of N}$
$5\text{ Improve the partition (N+}_c,\text{N-}_c)$
$\text{Return (N+,N-)}$
$\text{endif}$

cf. Multigrid method
Hypergraph-based Load Balancing

1. Hypergraph = ({node}, {hyperedge = a group of nodes})
2. More expressive power for computation-communication relation compared with graphs

U. V. Catalyurek et al., “Hypergraph-based dynamic load balancing for adaptive scientific computations,” in Proc. IPDPS (IEEE, ’07)

M. Kunaseth et al., “A scalable parallel algorithm for dynamic range-limited n-tuple computation in many-body molecular dynamics simulation,” in Proc. SC (ACM/IEEE, ’13)
Hybrid Decomposition

Aiichiro Nakano

Collaboratory for Advanced Computing & Simulations
Department of Computer Science
Department of Physics & Astronomy
Department of Chemical Engineering & Materials Science
University of Southern California

Email: anakano@usc.edu

Who does what?
**Pathways to a Protein Folding Intermediate Observed in a 1-Microsecond Simulation in Aqueous Solution**

Yong Duan and Peter A. Kollman*

An implementation of classical molecular dynamics on parallel computers of increased efficiency has enabled a simulation of protein folding with explicit representation of water for 1 microsecond, about two orders of magnitude longer than the longest simulation of a protein in water reported to date. Starting with an unfolded state of villin headpiece subdomain, hydrophobic collapse and helix formation occur in an initial phase, followed by conformational readjustments. A marginally stable state, which has a lifetime of about 150 nanoseconds, a favorable solvation free energy, and shows significant resemblance to the native structure, is observed; two pathways to this state have been found.

*Science* **282**, 740 ('98)

---

**Table 1**: NAMD performance on 327K atom ATPase benchmark system with and multiple timestepping with PME every four steps for Charm++ based on MPI and Elan.

<table>
<thead>
<tr>
<th>Processors</th>
<th>Time/step</th>
<th>Speedup</th>
<th>GFLOPS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total</td>
<td>Per Node</td>
<td>MPI</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td>28.08 s</td>
</tr>
<tr>
<td>128</td>
<td>4</td>
<td></td>
<td>248.3 ms</td>
</tr>
<tr>
<td>256</td>
<td>4</td>
<td></td>
<td>135.2 ms</td>
</tr>
<tr>
<td>512</td>
<td>4</td>
<td></td>
<td>65.8 ms</td>
</tr>
<tr>
<td>510</td>
<td>3</td>
<td></td>
<td>65.7 ms</td>
</tr>
<tr>
<td>1024</td>
<td>4</td>
<td></td>
<td>41.9 ms</td>
</tr>
<tr>
<td>1023</td>
<td>3</td>
<td></td>
<td>35.1 ms</td>
</tr>
<tr>
<td>1536</td>
<td>4</td>
<td></td>
<td>35.4 ms</td>
</tr>
<tr>
<td>1536</td>
<td>3</td>
<td></td>
<td>26.7 ms</td>
</tr>
<tr>
<td>2048</td>
<td>4</td>
<td></td>
<td>31.8 ms</td>
</tr>
<tr>
<td>1800</td>
<td>3</td>
<td></td>
<td>25.8 ms</td>
</tr>
<tr>
<td>2250</td>
<td>3</td>
<td></td>
<td>19.7 ms</td>
</tr>
<tr>
<td>2400</td>
<td>4</td>
<td></td>
<td>32.4 ms</td>
</tr>
<tr>
<td>2800</td>
<td>4</td>
<td></td>
<td>32.3 ms</td>
</tr>
<tr>
<td>3000</td>
<td>4</td>
<td></td>
<td>32.5 ms</td>
</tr>
</tbody>
</table>
FIG. 5. The division of the permuted force matrix $F'$ among 16 processors in the force-decomposition algorithm. Processor $P_6$ is assigned a sub-block $F'_6$ of size $N/\sqrt{P}$ by $N/\sqrt{P}$. To compute its matrix elements it must know the corresponding $N/\sqrt{P}$-length pieces $x_\alpha$ and $x'_\beta$ of the position vector $x$ and permuted position vector $x'$.

Hybrid Spatial+Force Decomposition

- Spatial decomposition of patches (localized spatial regions & atoms within)
- Inter-patch force computation objects assigned to any processor
- Message-driven object execution

Neutral Territory Decomposition

D. E. Shaw,
“A fast, scalable method for the parallel evaluation of distance-limited pairwise particle interactions,”

cf. Lecture note on “Shaw’s NT algorithm”