Multipol: A Distributed Data Structure Library for Irregular Applications

Kathy Yelick

U.C. Berkeley

Soumen Chakrabarti
Etienne Deprit
Eun-Jin Im
Jeff Jones
Arvind Krishnamurthy
Steve Steinberg
Chih-Po Wen
Jun Yang

Outline

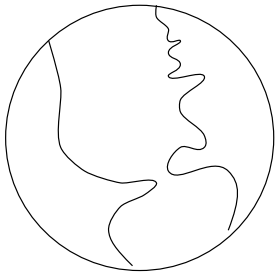
1. Irregular parallel programs

2. Three parallel programming problems

4. The Multipol Library

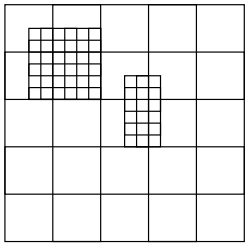
5. Open Problems (and Potential Class Projects)

The World is an Irregular Place

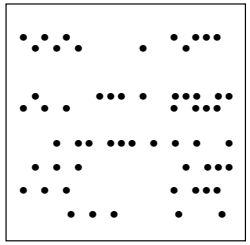


global climate modeling

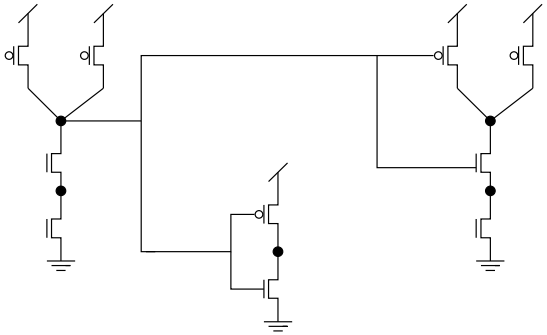
fluid flow



dynamic simulations need adaptive algorithms



most huge matrices are sparse



circuit simulation and verification

Sources of Irregularity

Control flow

```
for (i=0;i<n;i++)
  for (j=0;j<n;j++)
```

regular

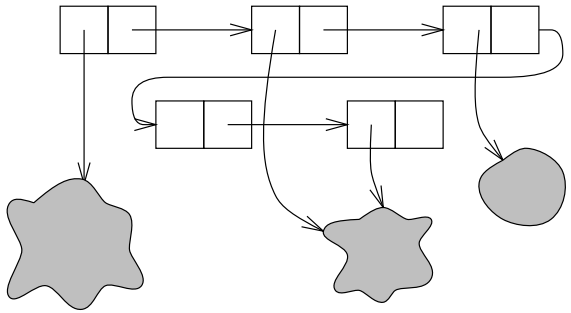
```
if (p(x))
   if (q(y))
```

irregular

Data Structures

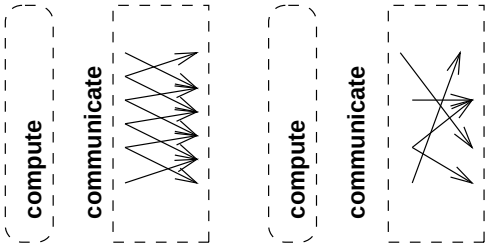
3	2	9	1	0
9	8	3	3	2
2	4	1	0	8
5	5	8	3	9
7	8	2	9	6

regular

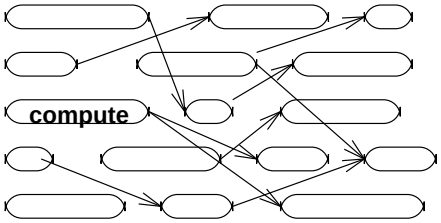


irregular

Communication



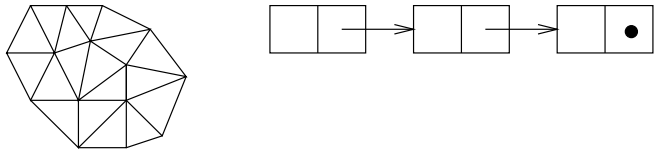
regular in space *regular in time*



irregular

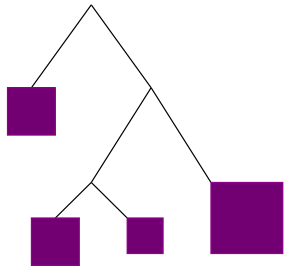
Three Parallel Programming Problems

Unstructured data



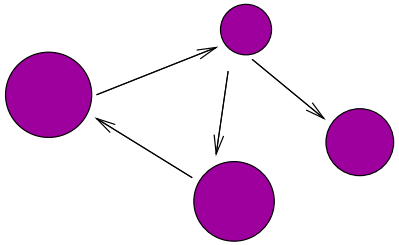
Global pointers in a 2D address space (processor/memory)

Load imbalance



Dynamic and static scheduling and load balancing

Fine-grained asynchronous communication



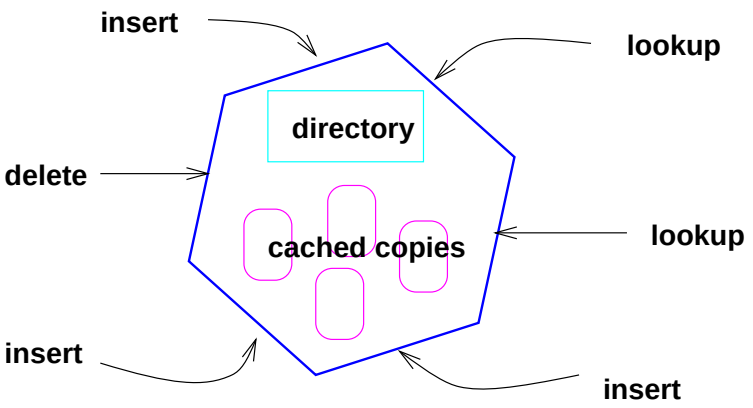
Aggregation to amortize overhead and encapsulation to hide details

Multipol Overview

Multipol Characteristics

Designs use:

- partitioning
- replication
- dynamic caching
- hierarchical storage



Programmer has two views:

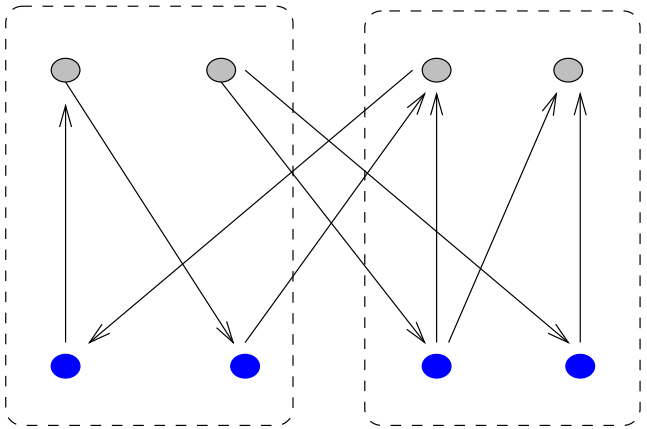
* global shared object
* local per processor portion

Common theme: locality and load balance tradeoff

Partitioning Irregular Data Structures

Irregular meshes may be partitioned across processors

- Graph mostly partitioned
- Replicate boundary values
- Lazy ghost node updates



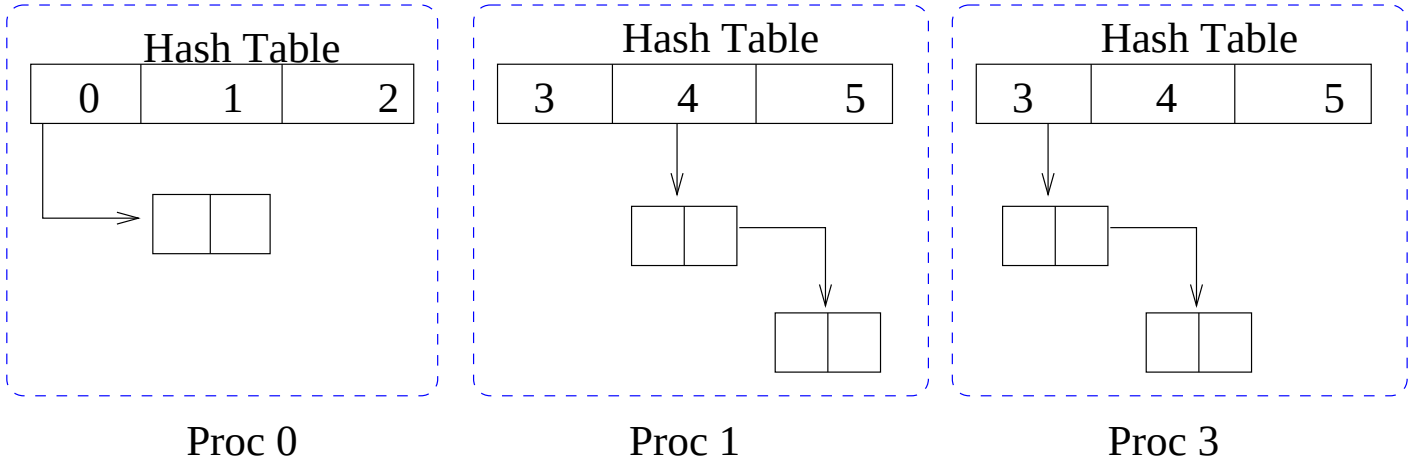
Bipartite graph avoids concurrency control problems: only one part is updated at a time

Two parts may both be irregular (EM3D)

Or one regular and one irregular (Cells)

Graph partitioning is an important partially-solved problem

Partitioned Hash Table



Characteristics:

Single message to access most values

Hash function provides load balance.

Examples:

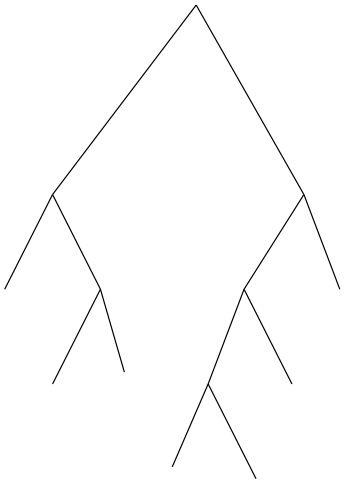
Used in to store solutions in search

Used in n-body problems

Load Imbalance

In sequential programs, stacks, queues, and priority queues are used to hold tasks.

In parallel programs these should be done in parallel.



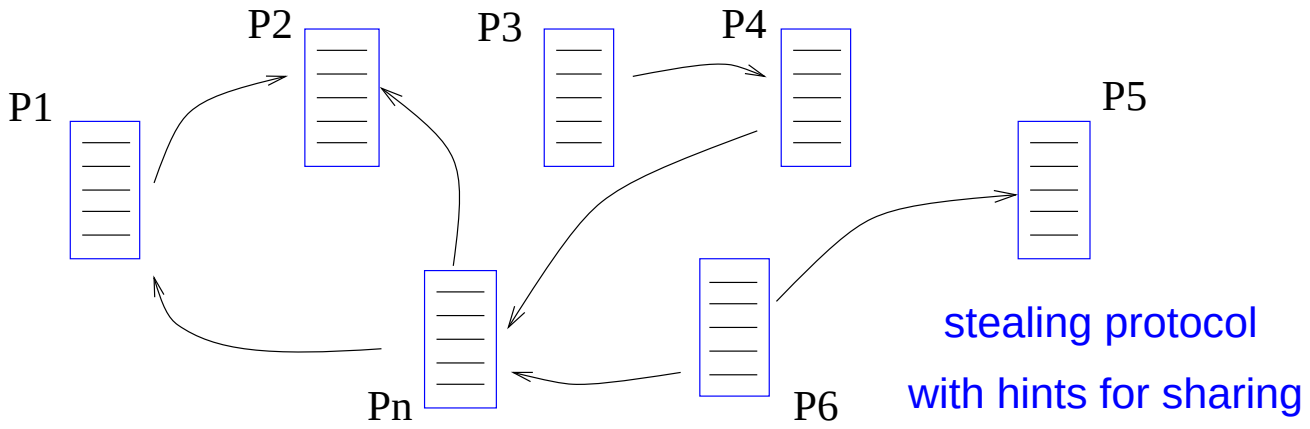
Both shape and weight of tree may vary

Split-C's SPMD model doesn't match this dynamic

Approach 1: use a programming language with dynamic threads

Approach 2: provide a parallel data structure for load balancing tasks

Task Queue for Dynamic Load Balance



Characteristics:

- Mostly partitioned
- No strict priority or FIFO ordering
- Termination detection built in
- Locality heuristics

Future work

- Scheduling data-parallel tasks
- Improved locality

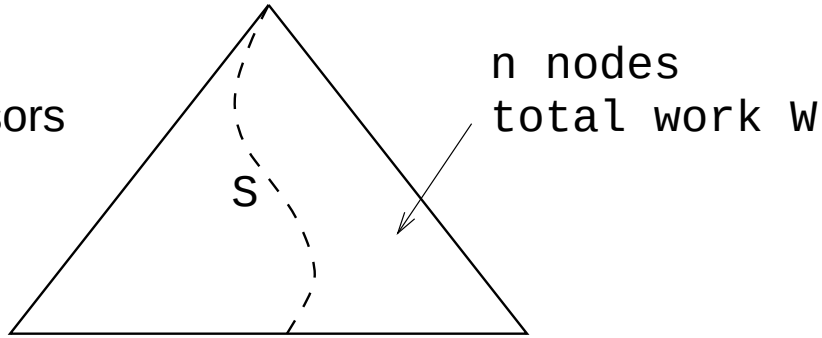
Load Balancing

Goals

Maximize processor utilization
Minimize centralization bottleneck

Model

Sequential running time = W
Parallel running time with P processors
> max { W/P, S } $=\Omega$ (W/P+S)



Application

Branch and bound
Divide and conquer