The First Year

A VALIDATION OF THE PROGRAMMING MODEL

John Feo
Tera Computer Company
Seattle WA 98102

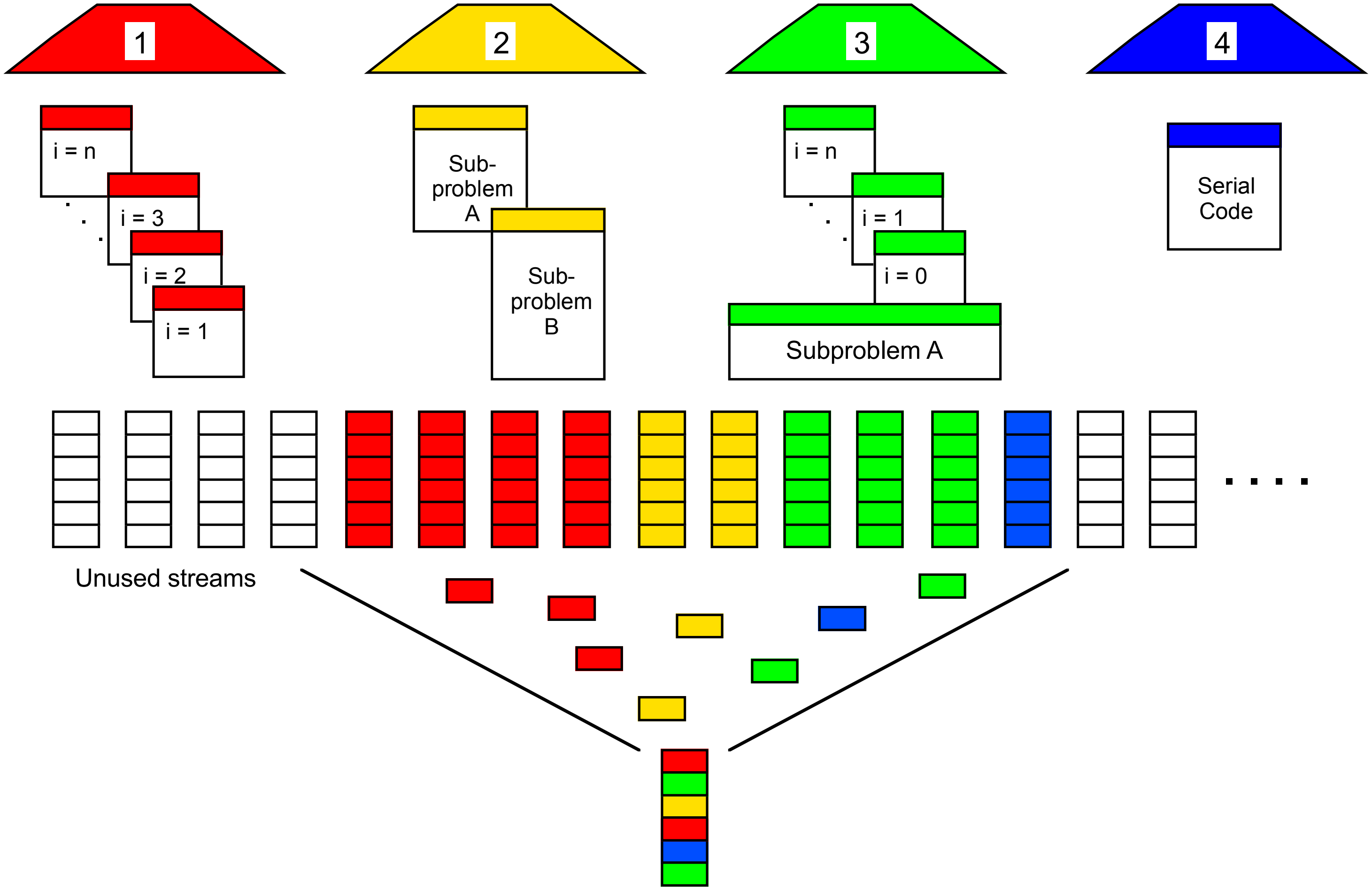
ORAP
Paris, France

October 20, 1998

Outline

- Current system
- User community
- Developing new methods
- Porting sequential codes
- Porting vector codes
- Porting MPP codes

View of a MTA processor



Programs running in parallel

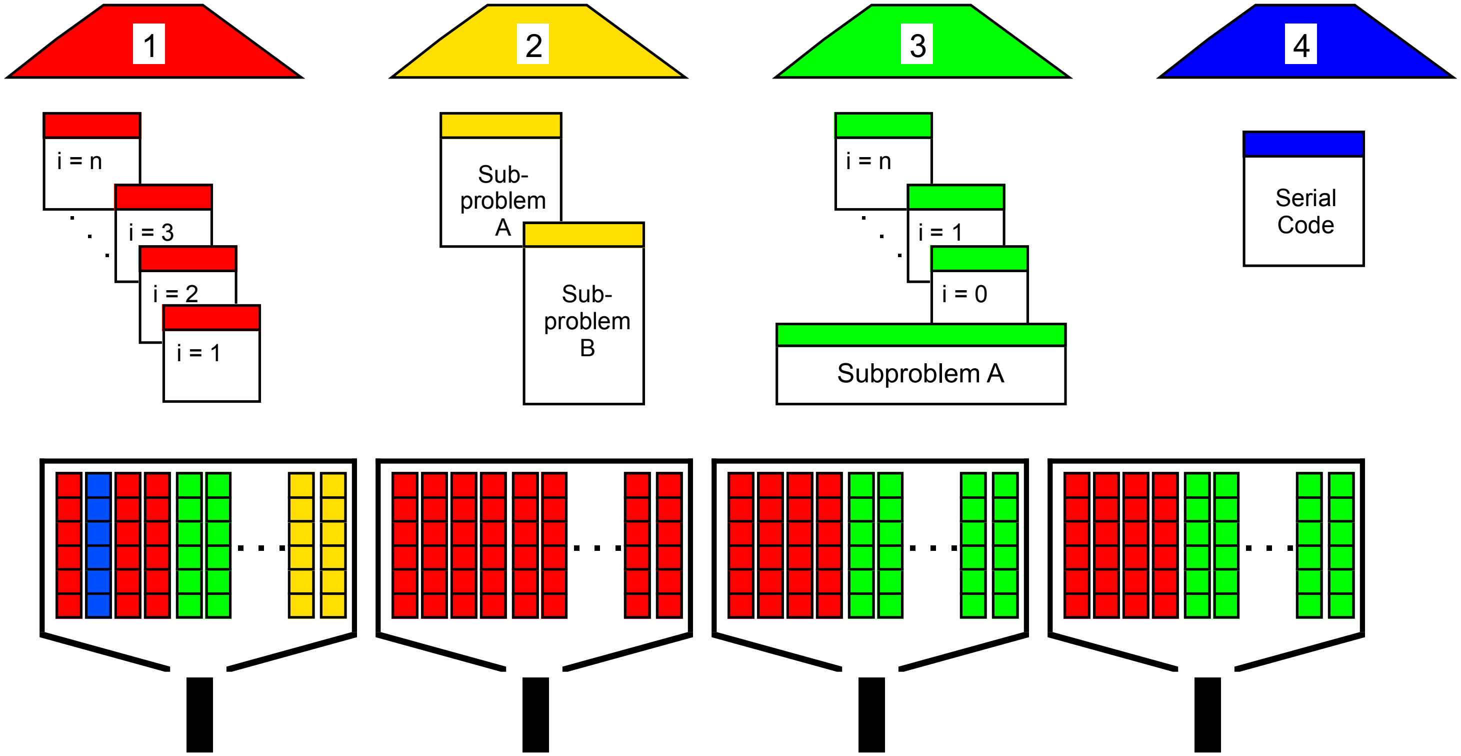
Concurrent threads of computation

Hardware streams

Instruction Ready Pool

Pipeline of executing instructions

View of multiple MTA processors



Programs running in parallel

Concurrent threads of computation

Multithreaded across multiple processors

Achieving high utilization

- Have enough active streams such that on every cycle at least one stream has a ready instruction
- Need at least the maximum of
  - instruction pipeline latency (21 cycles)
  - (memory latency) / (average lookahead)
- Typically, you need 40 to 80 streams per processor to achieve utilization greater than 90%

What is important

-
  ```
  Bad_Performance = (.not. Enough_Streams .or.
                     .not. Enough_Threads)
  ```

-
  ```
  Bad_Performance = (! Enough_Streams ||
                     ! Enough_Threads)
  ```

What is irrelevant

- cache misses
- poor data locality
- static assignment of tasks to processors
- poor communication to computation ratio
- parallelism that is too fine or too coarse to exploit effectively

Current hardware

- 2 resource modules
  - 2 processors (255 MHZ)
  - 2 I/O processors
  - 2 GB memory
- Full speed, 2 processor network
  - Asymmetric in z-direction
- Upgrade to 4 resource modules, full bandwidth network expected this quarter

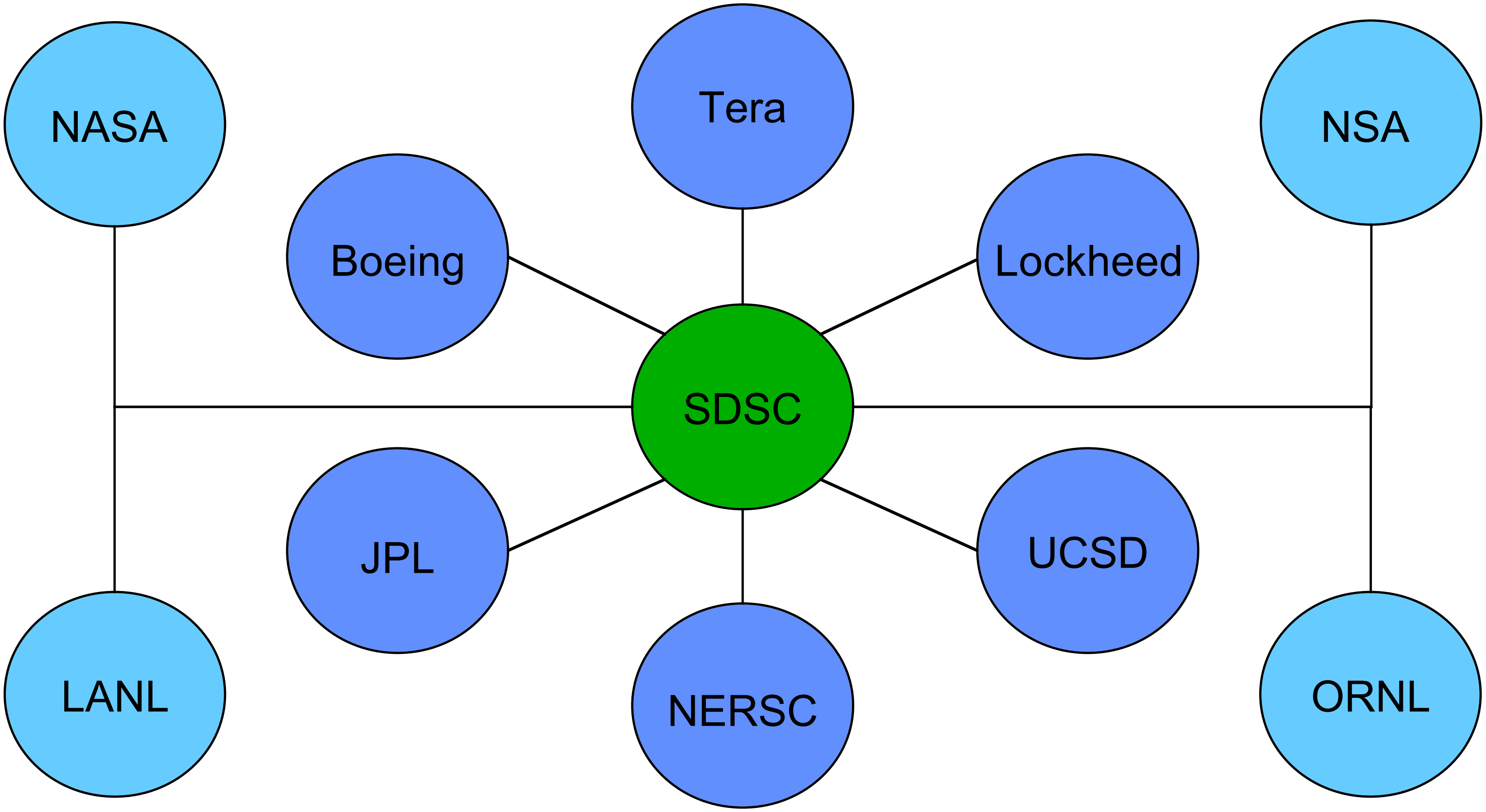
Current software

- MTX operating system
    - Most Unix utilities and capabilities
    - Ram file system
    - Sockets
- Fortran, C, and C++ compilers (including all libraries)
    - Canal
- TDB
- Traceview

Availability

- Hardware has been exceptionally stable and reliable
  - MTF many months
- MTX is immature, but reliable
  - MTF 3 - 4 hours
- Compilers, debugger, and performance analysis tools are mature
- Documentation is available at http://tera.sdsc.edu
  - (TERAreview, TERAreview)
- User accounts are controlled by SDSC

User community



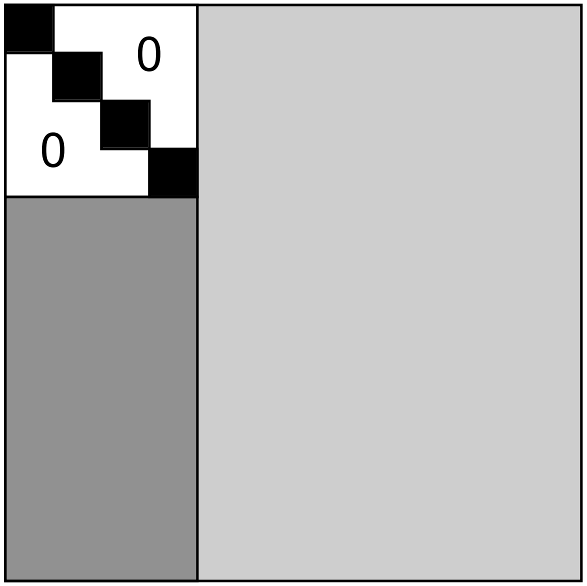
Developing new methods

- Sparse linear solvers are an important mathematical method
- Direct solvers are very difficult to parallelize on all MPP systems except the MTA
  - no data locality
  - fine-grain synchronization

Ncpus	T916/14-512 T90 Solver	MTA-2 Tera Solver
1	4.9 secs	5.2 secs
2	4.9 secs	3.3 secs

MTA Sparse Solver

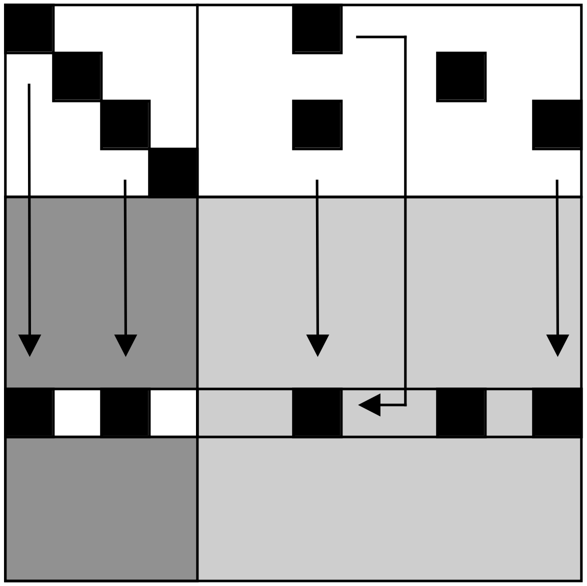
- Identify independent pivot rows (Luby’s algorithm)



MTA Sparse Solver (cont.)

- Reduce all pivot rows simultaneously

```
for each pivot row I
    for each non-pivot row j afftected by I
        for each non-zero element in i
```



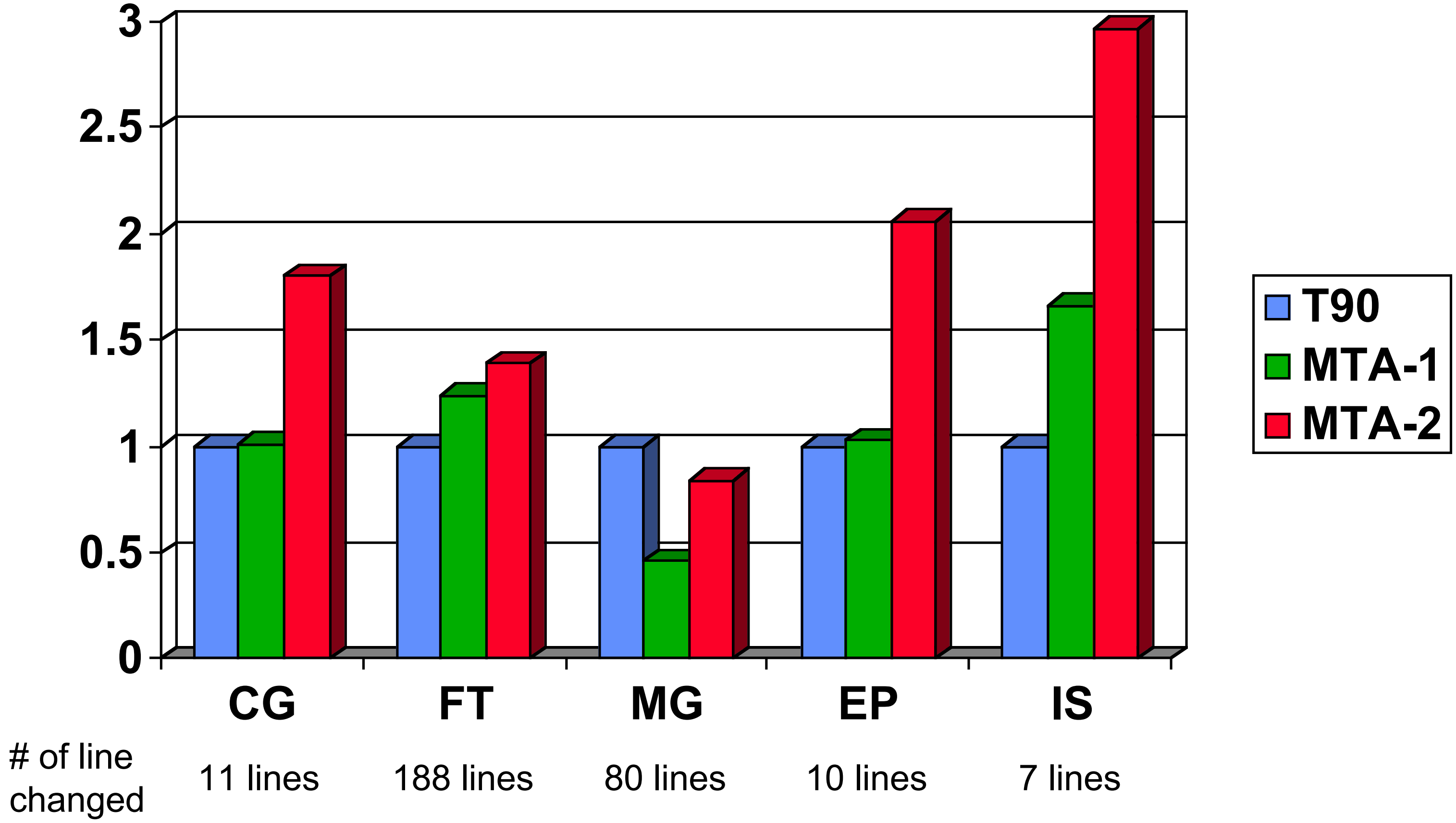
MTA Sparse Solver (cont.)

- Repeat steps 1 and 2
- As matrix fills in, the number of independent pivot rows decrease reducing the advantage of identifying independent sets
- Parallelism remains high throughout since the second and third loops increase in work as the first loop shrinks in work

Porting sequential codes

- Remove spurious dependencies
- Remove unnecessary data copying
  - No data cache
  - No stride sensitivity
- Re-write loops to operate over whole data structures
  - No blocking
  - No preferred direction
- Parallelize outer loops

NAS Parallel Benchmarks



Porting vector codes

- Multi-tasked vector codes are ideally suited for the MTA
  - our compiler recognizes all CRAY pragmas
- Parallelize multi-tasked loop
  - Decreasing “vector length” increases outer-level parallelism
- Remove critical sections protecting global updates and/or move updates into main computational loop; use synchronized loads/stores to assure correctness
- Break-up inner vector loops to increase opportunities for software pipelining

FEM3D on the T-90

```
cmic$ do all shared  (...)
cmic$        private (...)
       do 235 ielg = 1, nelg, ncpub

          do nnn = ielg, ielg + ncpub - 1
             if (ielg .le. nelgh) then
                call solde(...)
             elseif (ielg .le. nelgh) then
                call elem1d(...)
             elseif (ielg .le. nelgh) then
                call elem2d(...)
             else
                call thicks(...)
             endif

cmic$ guard
       ... release scratch space to next task
cmic$ end guard

  235 continue
```

Fem3D on the MTA

```
c$tera assert parallel
c$tera dynamic schedule
       do 235 ielg = 1, nelg
c$tera assert local (...)

          if (ielg .le. nelgh) then
             call solde(...)
          elseif (ielg .le. nelgh) then
             call elem1d(...)
          elseif (ielg .le. nelgh) then
             call elem2d(...)
          else
             call thicks(...)
          endif

  235  continue
```

DYNA3D performance

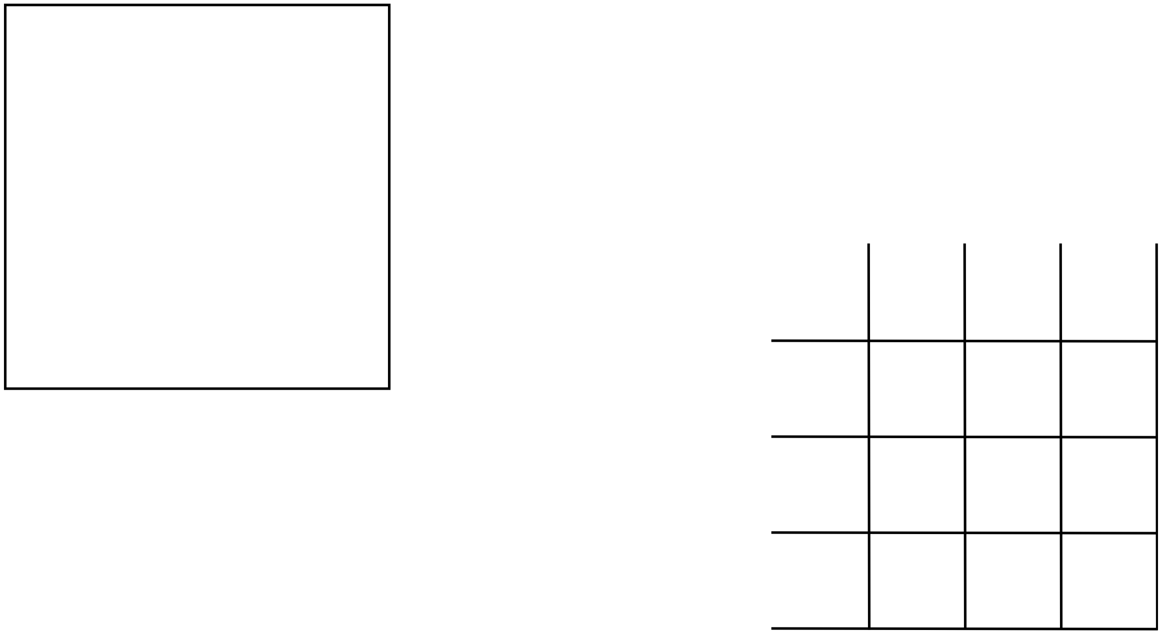
Benchmark	T90-1	T90-2	MTA-1	MTA-2
C2500 NJ Barrier	154	227	105	171

MFLops

Porting MPP codes

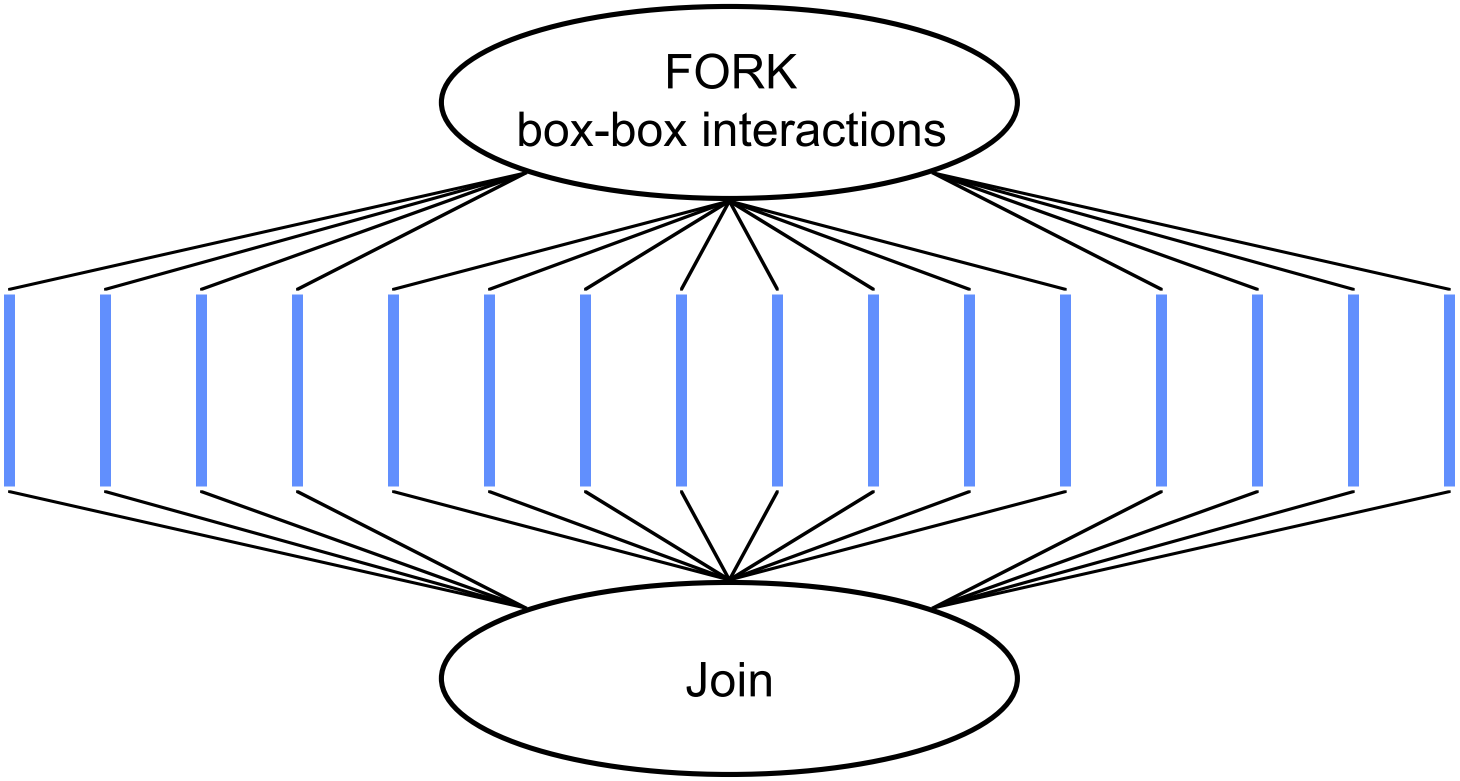
- Set number of processors to 1
- Remove all communications, data identification, data distribution, task scheduling, etc., etc.
- Adjust number of “system elements” to assure sufficient outer level parallelism
  - particles
  - orbital sets
  - zones of a grid
- Insert a loop over elements at fork-join points

MPP NWCHEM–MD



- Each processor is responsible for a set of
- If the dimensions of the boxes equal the cu
  nearest neighbor communications
- But fewer the boxes, greater the load imba

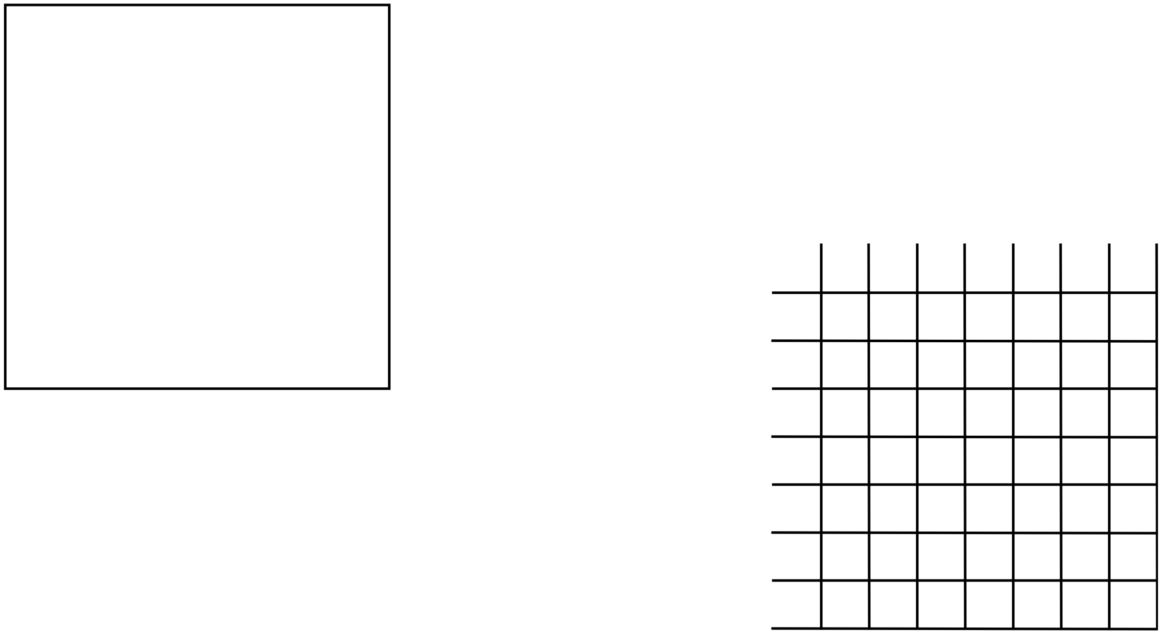
MPP code going parallel



FORK
box-box interactions

Join

MTA NWCHEM–MD



- Since there are no communication costs, s enough to insure lots of parallelism and g
- Insert loop over all box-box interactions at code

MTA code going parallel

```
c$tera assert parallel
c$tera dynamic schedule
       do 5 I = 1, nbbx
          … …
          … …
          … …
          … …
          … …
  5    continue
```