First experiences in hybrid parallel programming in quad-core Cray XT4 architecture

Sebastian von Alfthan and Pekka Manninen
CSC - the Finnish IT-center for science

CSC

Outline

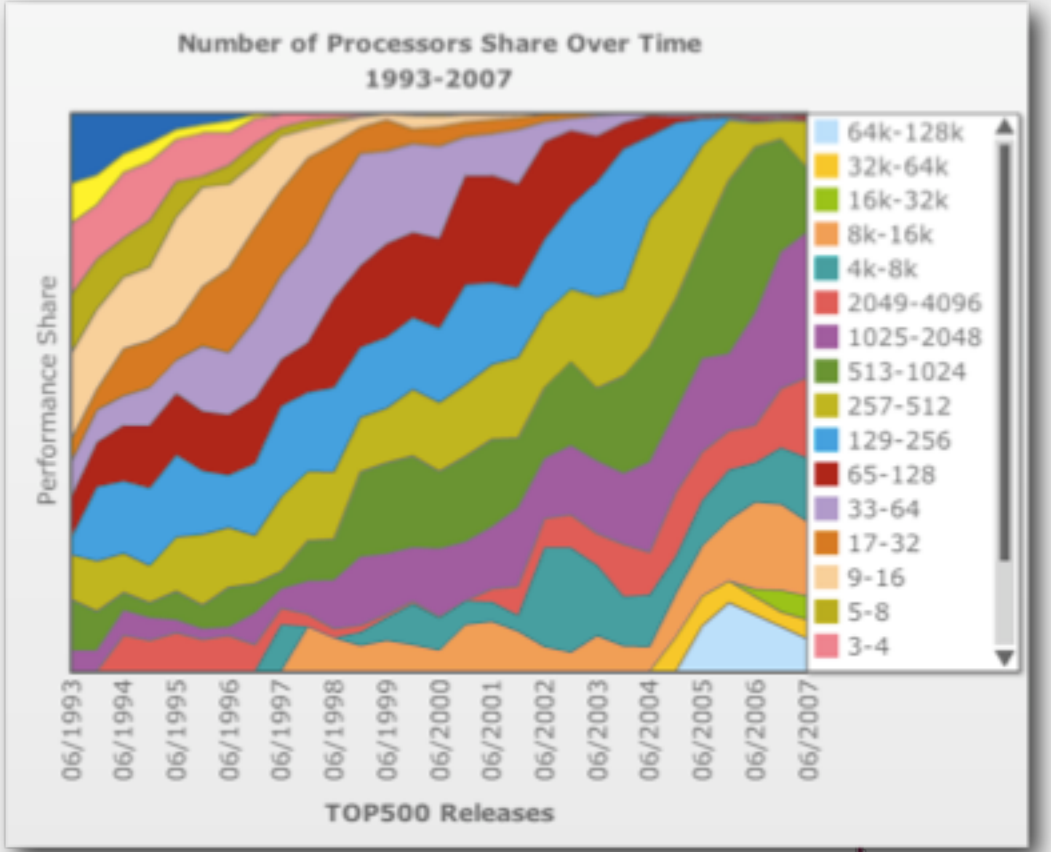
- **Introduction to hybrid programming**
- **Case studies**
  - Collective operations
  - Master slave algorithm
  - Molecular dynamics
  - I/O
- **Conclusions**

The need for improved parallelism

- **In less than ten years time every machine on Top-500 will be of peta-scale**
- **Free lunch is over, cores are not getting (very much) faster**
- **Achievable through a massive increase in the number of cores (and vector co-processors)**

The need for improved parallelism

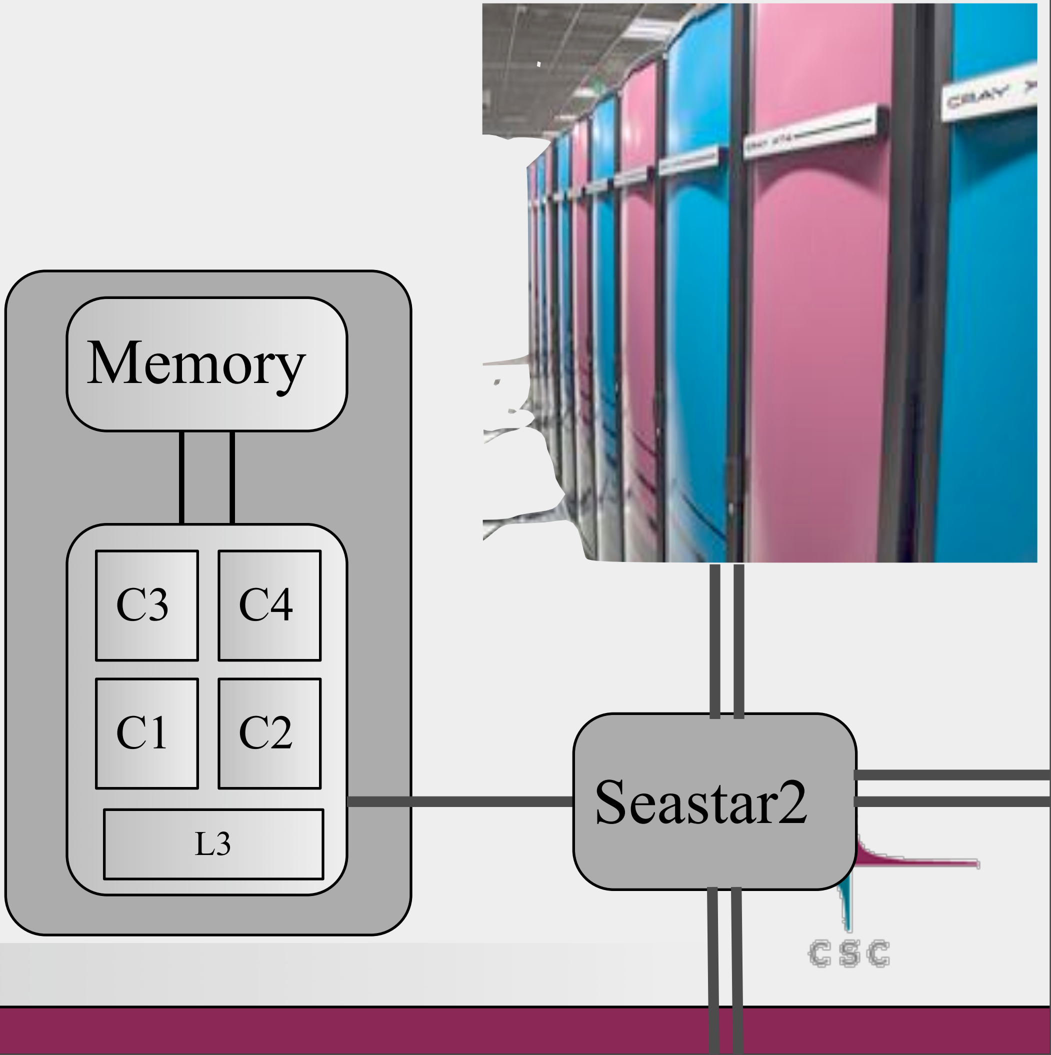
- In less than ten years time every machine on Top-500 will be of peta-scale
- Free lunch is over, cores are not getting (very much) faster
- Achievable through a massive increase in the number of cores (and vector co-processors)



CSC

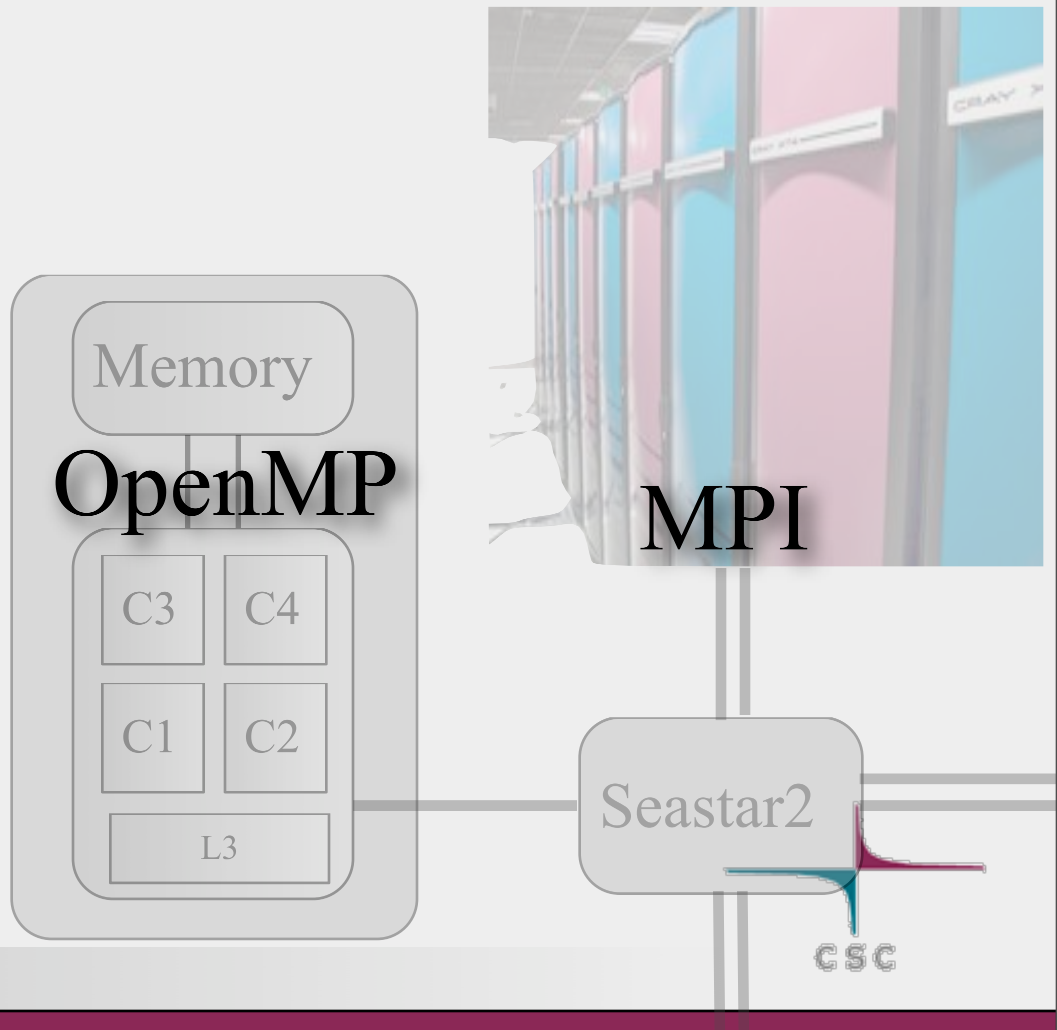
Cray - XT4

- Shared memory node with one quad-core Opteron (Budapest)
- Shared 2 MB L3 cache
- Memory BW 0.3 bytes/ flop
- Interconnect BW 0.2 bytes/flop
- How can we get good scaling with decreasing BW per flop?



Hybrid programming

- **Parallel programming model combining:**
- **OpenMP**
  - Shared memory parallelization
  - Directives instructing compiler on how to share data and work
  - Parallelization over one node
- **MPI**
  - Message passing library
  - Data communicated between nodes with messages



Expected benefits and problems

- \+ **Message aggregation and reduced communication**
- \+ **Intra-node communication is replaced by direct memory reads**
- \+ **Better load-balancing due to fewer MPI-processes**
- \+ **More options for overlapping communication and computation**
- \+ **Decreased memory-consumption**
- \+ **Improved cache-utilization, especially of shared L3**

- – **Difficult to code an efficient hybrid program**
- – **Tricky synchronization issues**
- – **Overhead from OpenMP parallelization**

OpenMP overhead

- **Thread management**
  - Creating/destroying threads
  - Critical sections
- **Synchronization**
- **Parallelism**
  - Imbalance
  - Limited parallelism
- **Overhead of for directive**
  - Avoid *guided* and *dynamic* unless necessary
  - Small loops should not be parallelized

	2 threads	4 threads
PARALLEL	0.5 µs	1.0 µs
STATIC(1)	0.9 µs	1.3 µs
STATIC(64)	0.4 µs	0.7 µs
DYNAMIC(1)	34 µs	315 µs
DYNAMIC(64)	1.2 µs	2.7 µs
GUIDED(1)	15 µs	214 µs
GUIDED(64)	3.3 µs	6.2 µs

Hybrid parallel programming models

1. **No overlapping communication and computation**
   1.1. MPI is called only outside parallel regions and by the master thread
   1.2. MPI is called by several threads
2. **Communication and computation overlap: while the some of the thread communicate, the rest are executing an application**
   2.1. MPI is called only by the master thread
   2.2. Communication is carried out with several threads
   2.3. Each thread handles its own communication demands

- **Implementation can further be categorized as**
  - **Fine-grained:** loop level, several local parallel regions
  - **Coarse-grained:** parallel region extends over larger segment

Hybrid programming on XT4

- **MPI-libraries can have four levels of support for hybrid programming**
- **MPI_THREAD_SINGLE**
  - Only one thread allowed
- **MPI_THREAD_FUNNELED**
  - Only master thread allowed to make MPI calls
  - Models 1.1 and 2.1
- **MPI_THREAD_SERIALIZED**
  - All threads allowed to make MPI calls, but not concurrently
  - Models 1.1 and 2.1, models 1.2, 2.2 and 2.3 with restrictions
- **MPI_THREAD_MULTIPLE**
  - No restrictions
  - All models

Hybrid programming on XT4

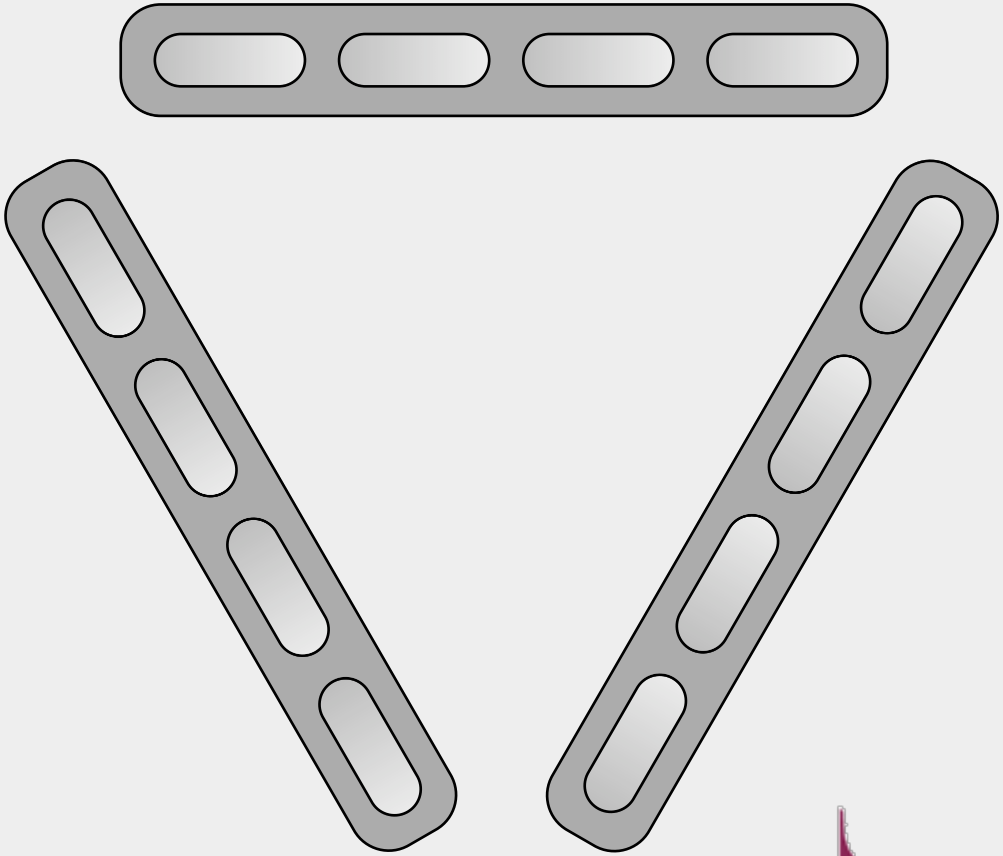
```
MPI_Init_thread(&argc,&argv,MPI_THREAD_MULTIPLE,&provided);

printf("Provided %d of %d %d %d %d\n",
provided,
MPI_THREAD_SINGLE,
MPI_THREAD_FUNNELED,
MPI_THREAD_SERIALIZED,
MPI_THREAD_MULTIPLE);

> Provided 1 of 0 1 2 3
```

Hybrid programming on XT4

- **MPI-library supports MPI_THREAD_FUNNELED**
- **Overlapping communication/computation still possible**
  - Non-blocking communication can be started in MASTER block
  - Completes while parallel region computes
- **Able to saturate the interconnect with only one thread communicating**

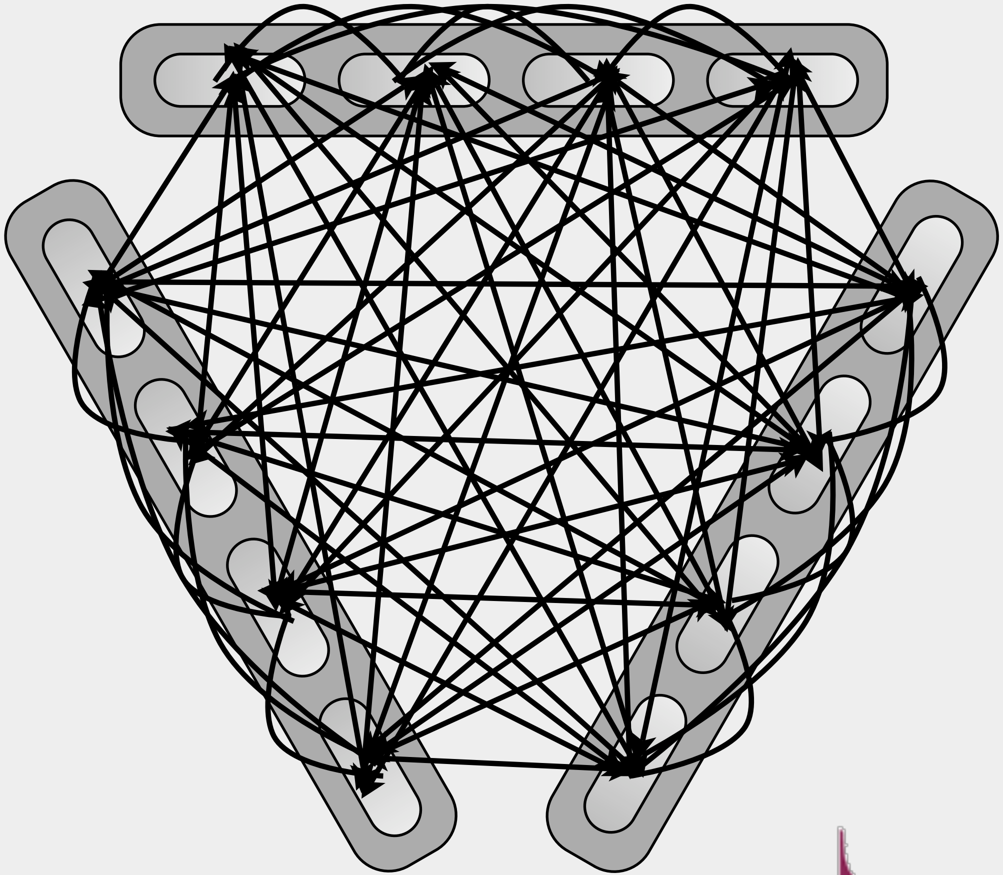
Case study 1: Collective operations

- **Collective operations often performance bottlenecks**
  - Especially all-to-all operations
  - Point-to-point implementation can be faster
- **Hybrid implementation**
  - For all-to-all operations (maximum) number of transfers decreases by a factor of #threads^2
  - Size of message increases by a factor of #threads
  - Allow overlapping communication and communication



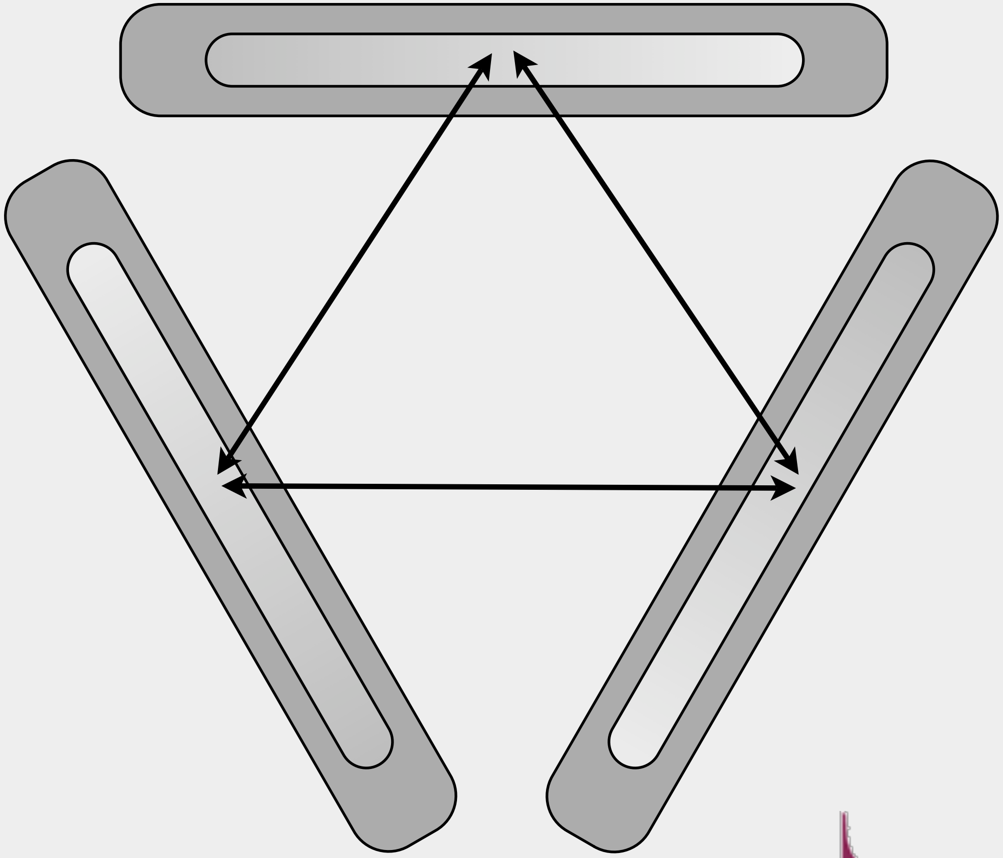
Case study 1: Collective operations

- **Collective operations often performance bottlenecks**
  - Especially all-to-all operations
  - Point-to-point implementation can be faster
- **Hybrid implementation**
  - For all-to-all operations (maximum) number of transfers decreases by a factor of #threads^2
  - Size of message increases by a factor of #threads
  - Allow overlapping communication and communication



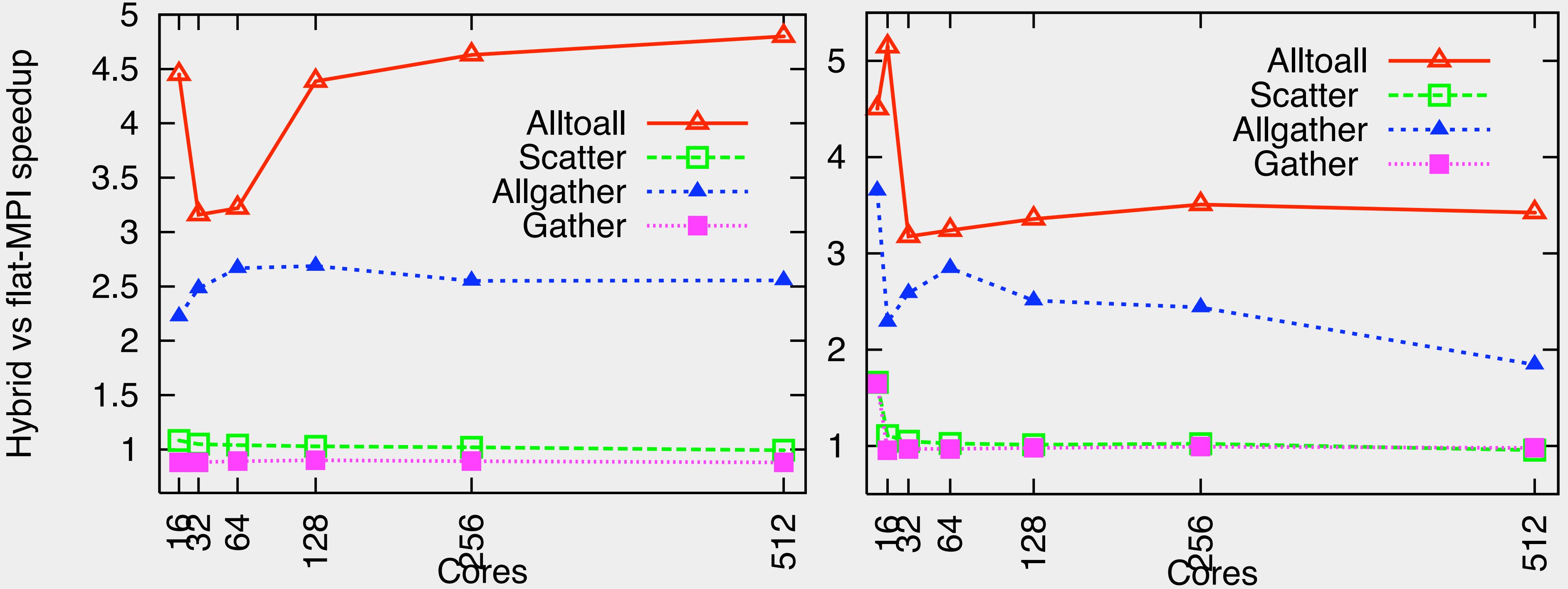
Case study 1: Collective operations

- **Collective operations often performance bottlenecks**
  - Especially all-to-all operations
  - Point-to-point implementation can be faster
- **Hybrid implementation**
  - For all-to-all operations (maximum) number of transfers decreases by a factor of #threads^2
  - Size of message increases by a factor of #threads
  - Allow overlapping communication and communication



CSC

Case study 1: Collective operations



40 Kbytes of data per node

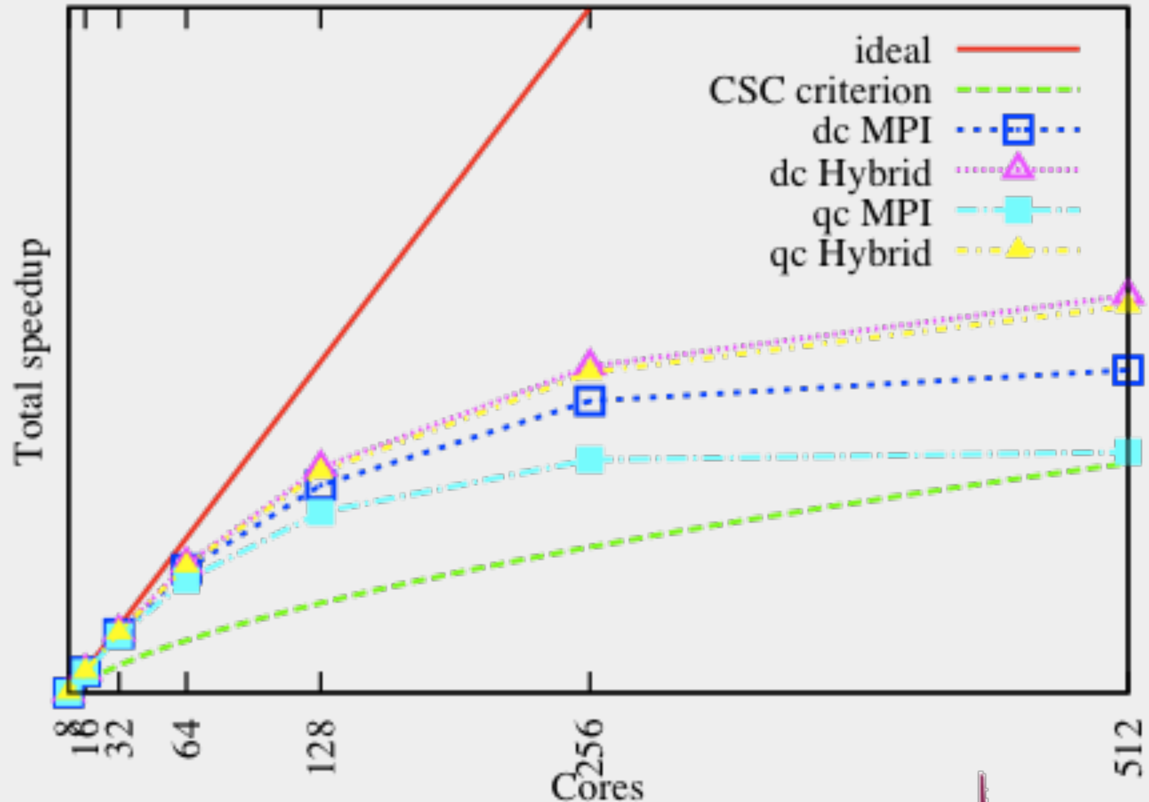
400 Kbytes of data per node

CSC

Case study 2: Master-slave algorithms

- ➢ **Matrix multiplication**
- ➢ **Demonstration of a master-slave algorithm**
- ➢ **Scaling is improved by going to a coarse-grained hybrid model**
- ➢ **Utilizes the following benefits:**

+ **Better load-balancing due to fewer MPI-processes**
+ **Message aggregation and reduced communication**

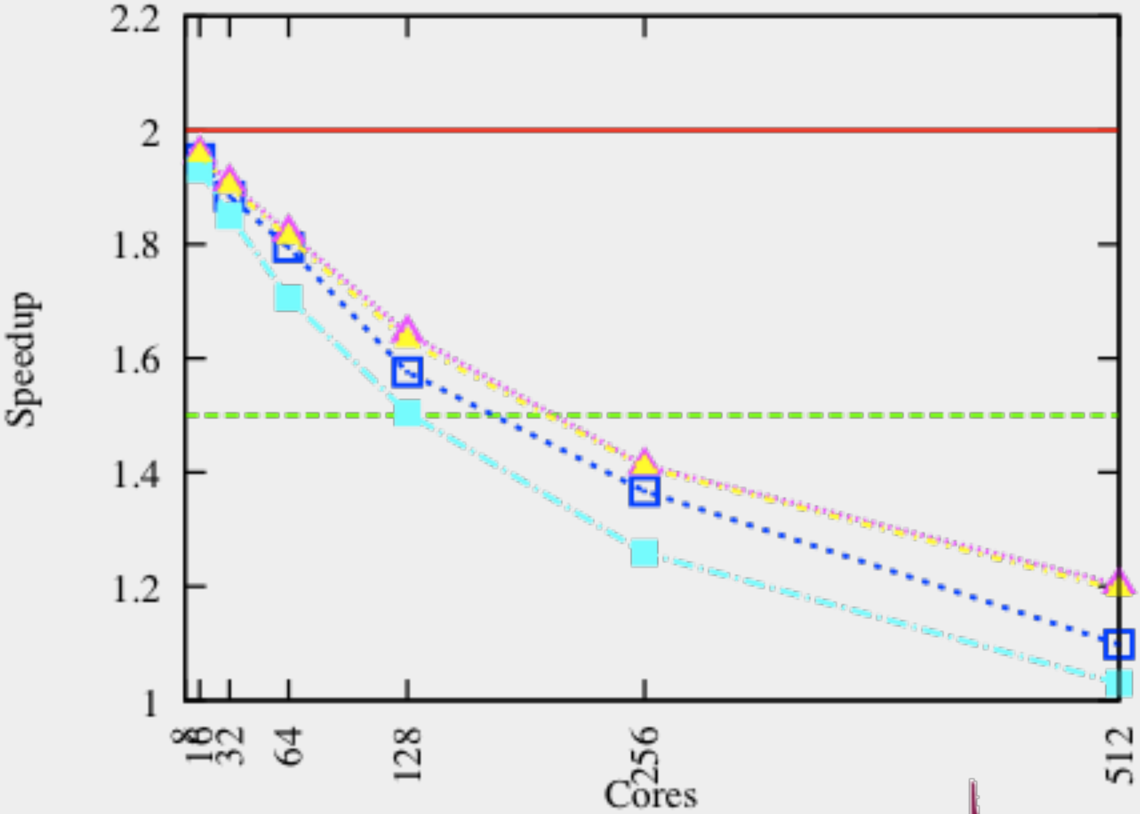


Case study 2: Master-slave algorithms

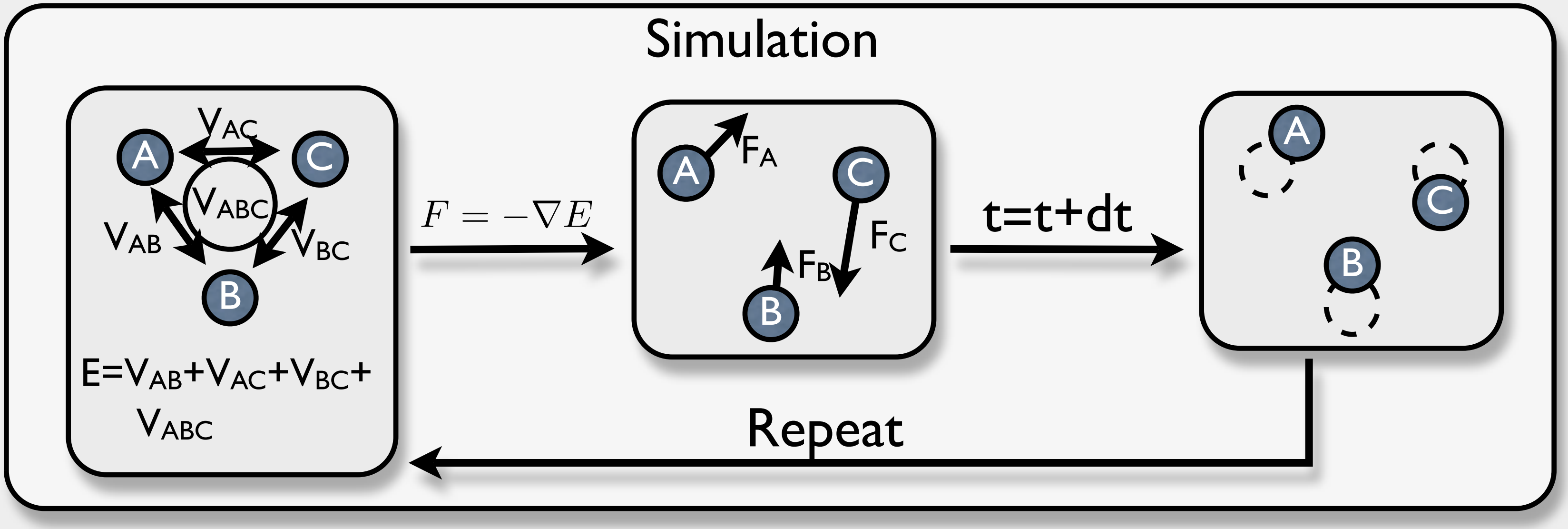
- Matrix multiplication
- Demonstration of a master-slave algorithm
- Scaling is improved by going to a coarse-grained hybrid model
- Utilizes the following benefits:

\+ Better load-balancing due to fewer MPI-processes

\+ Message aggregation and reduced communication



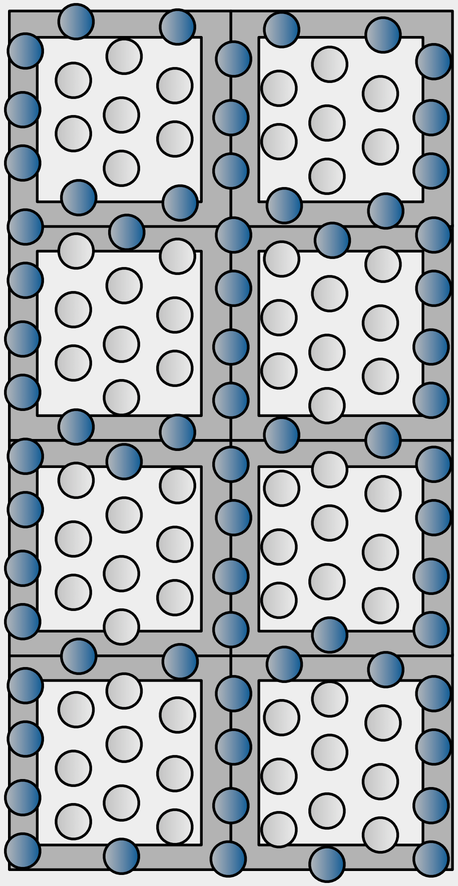
Case study 3: Molecular Dynamics



- Atoms are described as classical particles
- A potential model gives the forces acting on atoms
- Movement of atoms simulated by iteratively solving Newton's equations of motion

Case study 3: Domain decomposition

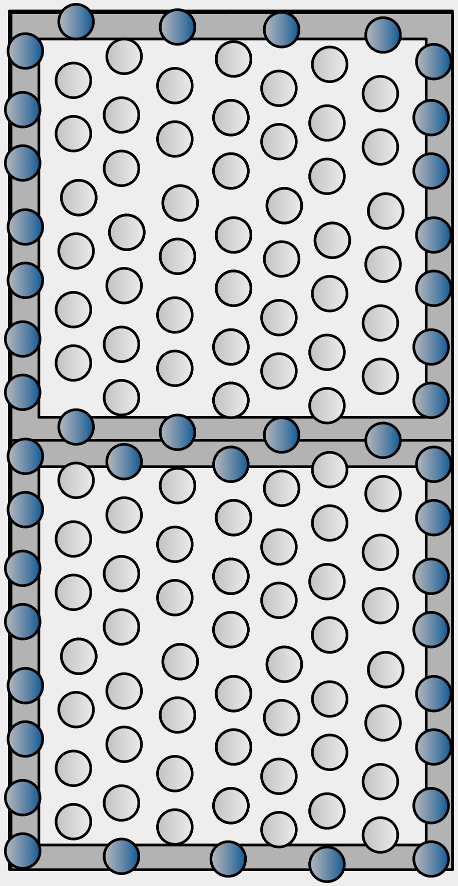
- **Number of atoms per cell is proportional to the number of threads**
- **Number of ghost particles is proportional to #threads$^{-1/3}$**
- **We can reduce communication by hybridizing the algorithm**
- **On quad-core the number of ghost particles decreases by about 40%**



CSC

Case study 3: Domain decomposition

- Number of atoms per cell is proportional to the number of threads
- Number of ghost particles is proportional to #threads$^{-1/3}$
- We can reduce communication by hybridizing the algorithm
- On quad-core the number of ghost particles decreases by about 40%



Case study 3: Molecular Dynamics

- **We have worked with Lammps**
  - Lammps is a classical molecular dynamics code
  - 125K lines of C++ code
  - http://lammps.sandia.gov/
- **"Easy" to parallelize length-scale (weak scaling)**
- **Time-scale difficult (strong scaling)**
  - Need a sufficient number of atoms per processor
- **Can we improve the performance with an hybrid approach?**
- **We have hybridized the Tersoff potential model**
  - Short-ranged
  - Silicon, Carbon...

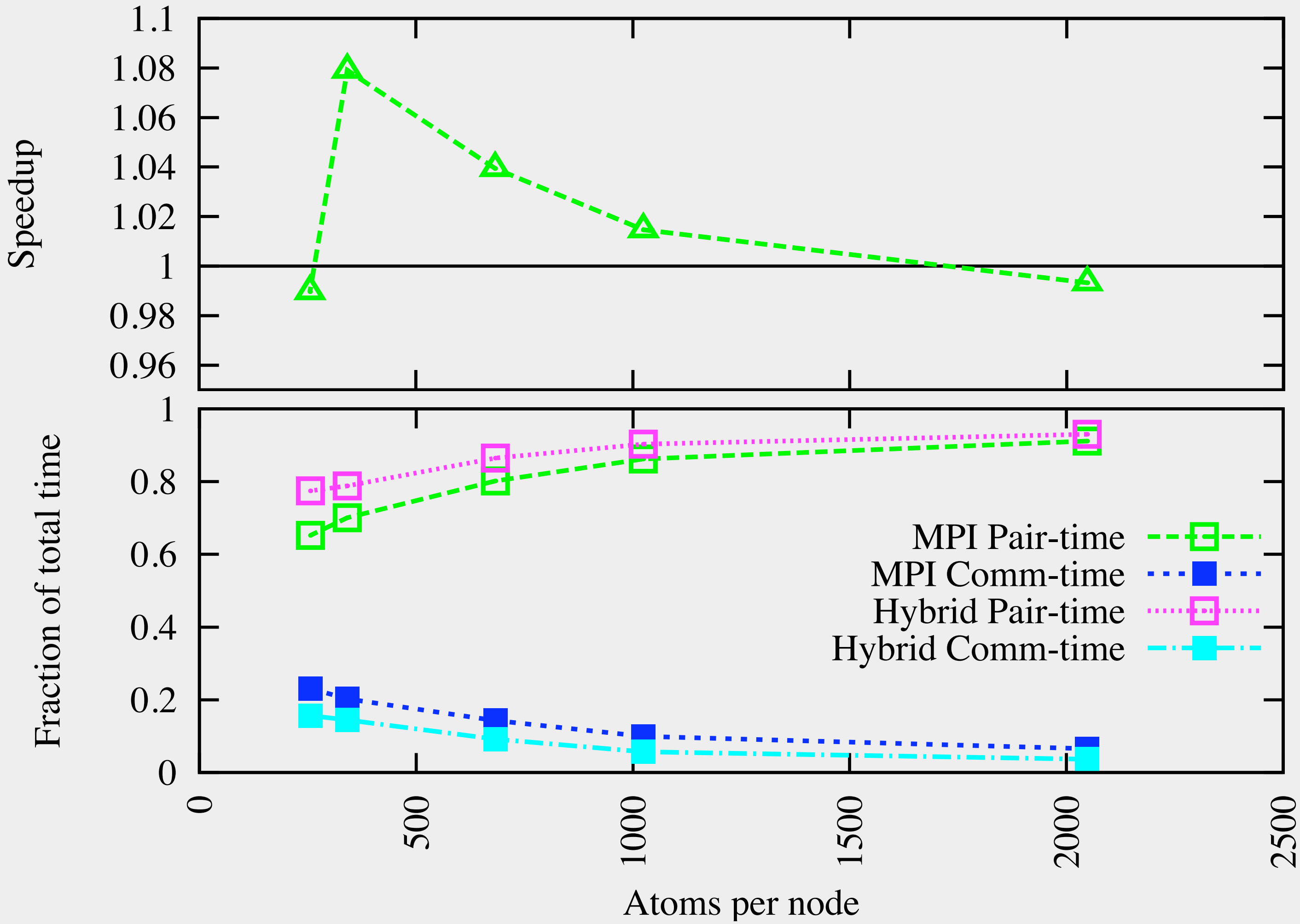
Case study 3: Algorithm

- **Fine-grained hybridization**
- **Parallel region entered each time the potential is evaluated**
- **Loop over atoms parallelized with static for**
- **Temporary array for forces**
  - Shared
  - Separate space for each thread
  - Avoids the need for synchronization when Newton's third law is used
  - Results added to real force array at end of parallel region

```
#pragma omp parallel
{
...
zero(ptforce[thread][..][..])
....
#pragma omp for schedule(static,1)
 for (ii = 0; ii < atoms; ii++)
  ...
  ptforce[thread][ii][..]+=....
  ptforce[thread][jj][..]+=....
}
...
for(t=0;t<threads;t++)
  force[..][..]+=ptforce[t][..][..]
...
```

CSC

Case study 3: Results for 32k atoms



Case study 3: Conclusions

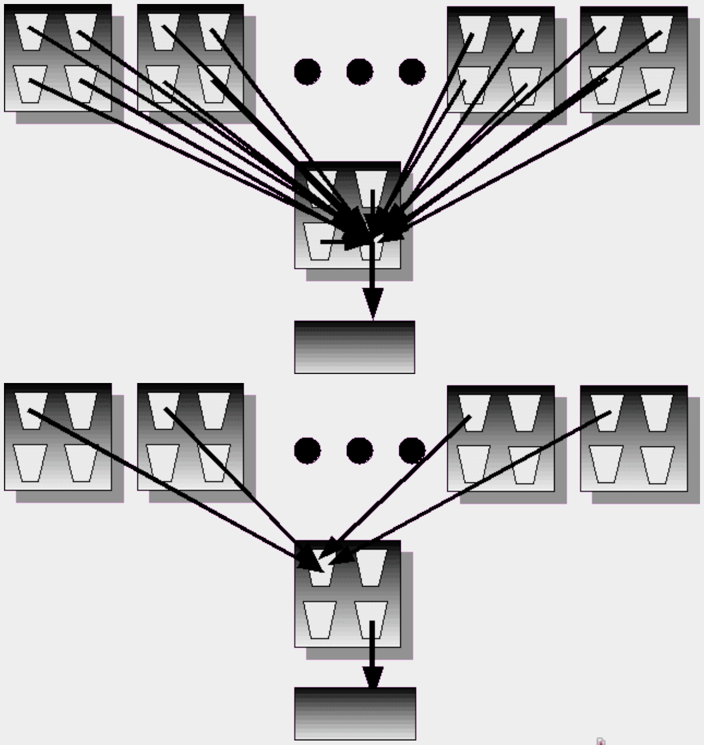
- **Proof-of-concept implementation**
- **Performance is**
  - **Improved** by decreased communication costs
  - **Decreased** by overhead in the potential model
- **Is there room for improvement..?**
  - Neighbor list calculation not parallelized
  - Coarse grained approach instead of fine grained
  - Other potential models have more communication (longer cut-off)

Case study 4: Parallel I/O

- **I/O is expensive and it is difficult to make it optimal**
- **Some approaches for parallel I/O**
  - Single writer reduction
  - MPI-2 I/O
  - N writers/readers to N files
  - Subset of writers/readers (J. Larkin CUG 2007)
- **We shall investigate these as implemented with flat MPI and by hybrid MPI/OpenMP**

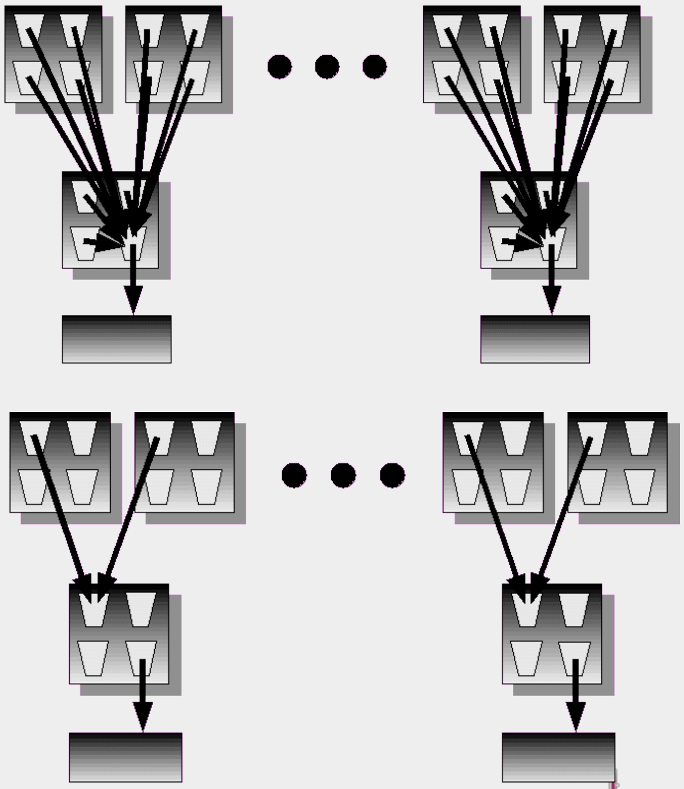
Case study 4: Single writer reduction

- All MPI processes send to one node for output
- Hybridization: only one core per processor sends a shared data array, on that node one core communicates, one writes (the rest may calculate)
- Bandwidth-limited



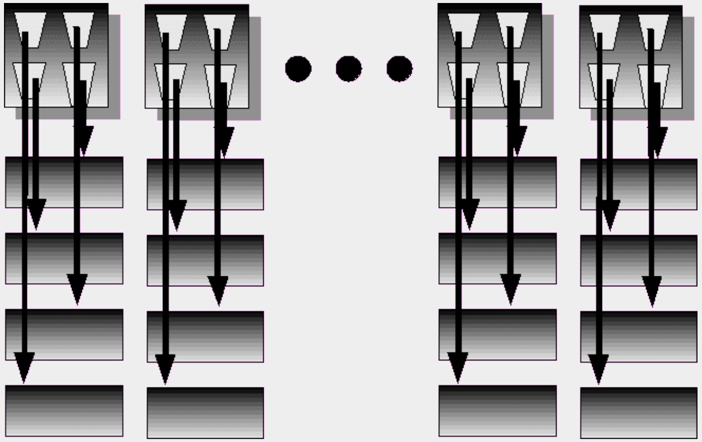
Case study 4: Subset of writers

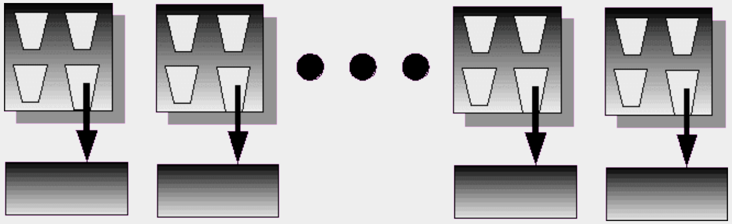
- Choose a subset of MPI tasks to do I/O, processes send their data to one of them
- Hybrid: one core communicates, one (or more) writes
- The optimal number of I/O nodes is not easy



Case study 4: N writers to N files

- **Every MPI process opens a file**
  - Good I/O BW
  - No communication needed
  - Large filesystem stress, slow open/closes
  - Inconvenient as many files are created
- **Hybridization: only one core per processor writes a shared array**
  - Achievable BW similar
  - Decreases number of files by a factor of #threads
  - Easy to implement
  - Allows overlapping of communication/computation





Conclusions

- **Hybrid approach is difficult, but sometimes useful**
- **Performance of hybrid approach is a tradeoff between greater overhead and decreased communication costs**
- **Direct benefits achieved without additional effort**
  - All-to-all collective operations 2-5 times faster
  - Gives parallel IO with reduced file-system stress in the N-writers case
  - Message aggregation
- **We expect the potential benefits to be even greater on XT5**

CSC

Acknowledgments

- **Cray inc.**
  - John Levesque and Jeff Larkin
  - Access to quad core XT4
- **Tekes**
  - FINHPC project

CSC