Februrary, 2003



Diskless Checkpointing

Applied to the Fast Fourier Transform

Christian Engelmann, Al Geist Oak Ridge National Laboratory

Super-scale Architectures

- Current tera-scale supercomputers have up to <u>10,000</u> processors.
- Next generation peta-scale systems will have <u>100,000</u> processors and more.
- Such machines may easily scale up to <u>1,000,000</u> processors in the next decade.
- IBM currently builds the BlueGene\L at Lawrence Livermore National Laboratory.

IBM BlueGene\L at LLNL

◆ Up to 64K diskless nodes with 2 processors per node. Only 256MB RAM per processor. Svstem 64 cabinets. 64x32x32 ♦ Additional service nodes (I/O). Cabinet Estimated 360 Tera FLOPS. (32 Node boards, 8x8x16) Node Board Over 150k processors. (32 chips, 4x4x2) 16 Compute Cards Compute Card Global tree network. (2 chips, 1x2x1) Chip 3-D torus network. 180/360 TF/s (2 processors) 16 TBDDR 29/5.7 TF/s 256 GB DDR Gigabit Ethernet. 90/180 GF/s 8 GB DDR 5.6/11.2 GE/s ♦ Operational in <u>2005</u>. 28/5.6 GF/s 0.5 GBDDR 4MB

Scalability Issues

How to make use of 100,000 processors?
 System scale jumps by a magnitude.
 Current algorithms do not scale well on existing 10,000-processor systems.

Next generation peta-scale systems are useless if efficiency drops by a magnitude.



Fault-tolerance Issues

How to survive on 100,000 processors?

- ♦ Failure rate grows with the system size.
- Mean time between failures may be a few hours or just a few minutes.
- Current solutions for fault-tolerance rely on checkpoint/restart mechanisms.
- Checkpointing 100,000 processors to central stable storage is not feasible anymore.

ORNL/IBM Collaboration

- Development of biology and material science applications for super-scale systems.
- Exploration of super-scalable algorithms.
 - Natural fault-tolerance.
 - Scale invariance.
- Focus on test and demonstration tool.
- Get scientists to think about scalability and fault-tolerance in super-scale systems!

Cellular Architecture Simulator

- Developed at ORNL in Java with native C and Fortran application support using JNI.
- Runs as standalone or distributed application.
- Lightweight framework simulates up to 1,000,000 processes on 9 real processors.
- Standard and experimental networks:
 - Multi-dimensional mesh/torus.
 - Nearest/Random neighbors.

Message driven simulation is not in real-time.
 Primitive fault-tolerant MPI support.



Super-scalable Fault-tolerance

- For non-naturally fault tolerant algorithms.
 Does it makes sense to restart all 100,000 processors because one failed?
- The mean time between failures is likely to be a few hours or just a few minutes.
- Traditional centralized checkpointing is limited by bandwidth (bottleneck).
- The failure rate is going to outrun the recovery and the checkpointing rate.

Diskless Checkpointing

Decentralized peer-to-peer checkpointing.
 Processors hold backups of neighbors.
 Local checkpoint and restart algorithm.
 Coordination of local checkpoints.





Diskless Checkpointing

In case of a failure:

- Rollback to local memory backup if necessary.
- Restart from remote memory backup.
- Encoding semantics, such as RAID, trade off storage size vs. degree of fault tolerance.
- Very infrequent checkpointing to central stable storage (disk/tape).
- Checkpoint and application processes may be the same or different.
- Possible OS support via library/service.

Choosing Neighbors

Physically near neighbors: Low latency, fast backup and recovery. Physically far neighbors: Recoverable multiprocessor node failures. Random neighbors: Medium latency and bandwidth. Acceptable backup and recovery time. Optimum: Pseudorandom neighbors based on system communication infrastructure.

Backup Coordination

- All peer-to-peer checkpoints need to be consistent with the global application state.
- Includes local states and in-flight messages.
- No backup coordination for checkpoints with no communication since the last one or start.
- Coordination techniques:
 - Global synchronization.
 - Local synchronization.

Global Synchronization

Global application snapshot (e.g. barrier) at stable global application state.

- Synchronous backup of all local states.
- Synchronizes complete application.
- Preferred method for communication intensive applications.
- Easy to implement.

Local Synchronization

- Asynchronous backup of local state and inflight messages (extensive message logging). Acknowledgements for messages to keep accurate records of in-flight messages. Additional local group communication. Different methods to retrieve missed messages from neighbors (replay/lookup). Preferred method for less communication intensive applications.
- More complicated to implement.

Application to FFT

Distributed and transposed FFT:

- Not naturally fault-tolerant.
- Every process is important.
- Not scale invariant.
- Mixture of local and global communication.
- Well known algorithm behavior.
- Other Fourier transform algorithms may be naturally fault-tolerant or scale better.
- They are not considered here.

How to checkpoint DFFT?

- Individual checkpoints with no synchronization.
 Coordinated checkpoints with global sync. due to heavy message load.
- Number of coordinated checkpoints depends on coefficients/processor.



How to checkpoint TFFT?

- Individual checkpoints with no synchronization.
 Coordinated checkpoints with local sync. due to light message load.
- Coordinated checkpoints only after transpose.
- More efficient than DFFT.



Observations

- Diskless peer-to-peer checkpointing on superscale architectures is possible.
- Synchronization methods have different strengths and weaknesses.
- Timing, latency and bandwidth data impossible to obtain from simulator.
- Real-time tests with different applications are needed for further discussion.
- Final real-world implementation requires super-scalable FT-MPI or PVM.

Conclusions

Super-scale systems with 100,000 and more processors become reality very soon.

- Diskless peer-to-peer checkpointing provides an alternative to natural fault-tolerance.
- ♦ A lot of research still needs to be done.

Februrary, 2003



Diskless Checkpointing

Applied to the Fast Fourier Transform

Christian Engelmann, Al Geist Oak Ridge National Laboratory