Diskless Checkpointing on Super-scale Architectures

Applied to the Fast Fourier Transform

Christian Engelmann, Al Geist
Network and Cluster Computing Group
Computer Science and Mathematics Division
Oak Ridge National Laboratory, Oak Ridge, USA

Overview

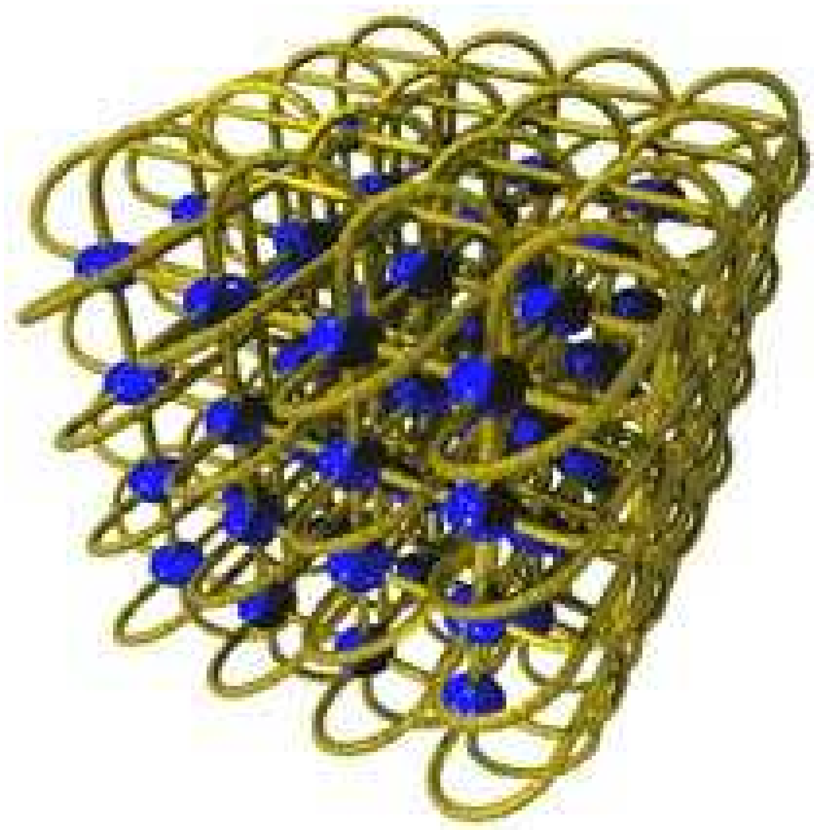
- Super-scale architectures.
- ORNL/IBM collaboration.
- Diskless checkpointing.
- Super-scale diskless checkpointing.
- Applied to the FFT.
- Conclusions.

Super-scale Architectures

- The current HPC tera-scale computers have up to 10,000 processors.
- The next generation peta-scale systems will have 50,000-100,000 processors.
- They will be deployed in the next 5 years.
- In the next decade, such machines may easily scale up to 1,000,000 processors.
- In 2002 IBM announced the Blue Gene/L.

IBM Blue Gene/L

- 65,536 processors.
- 256MB RAM/proc.
- 360 tera FLOPS.
- 3-D torus network.
- Tree network.
- Barrier network.
- Gigabit Ethernet.
- Operational: 2005.



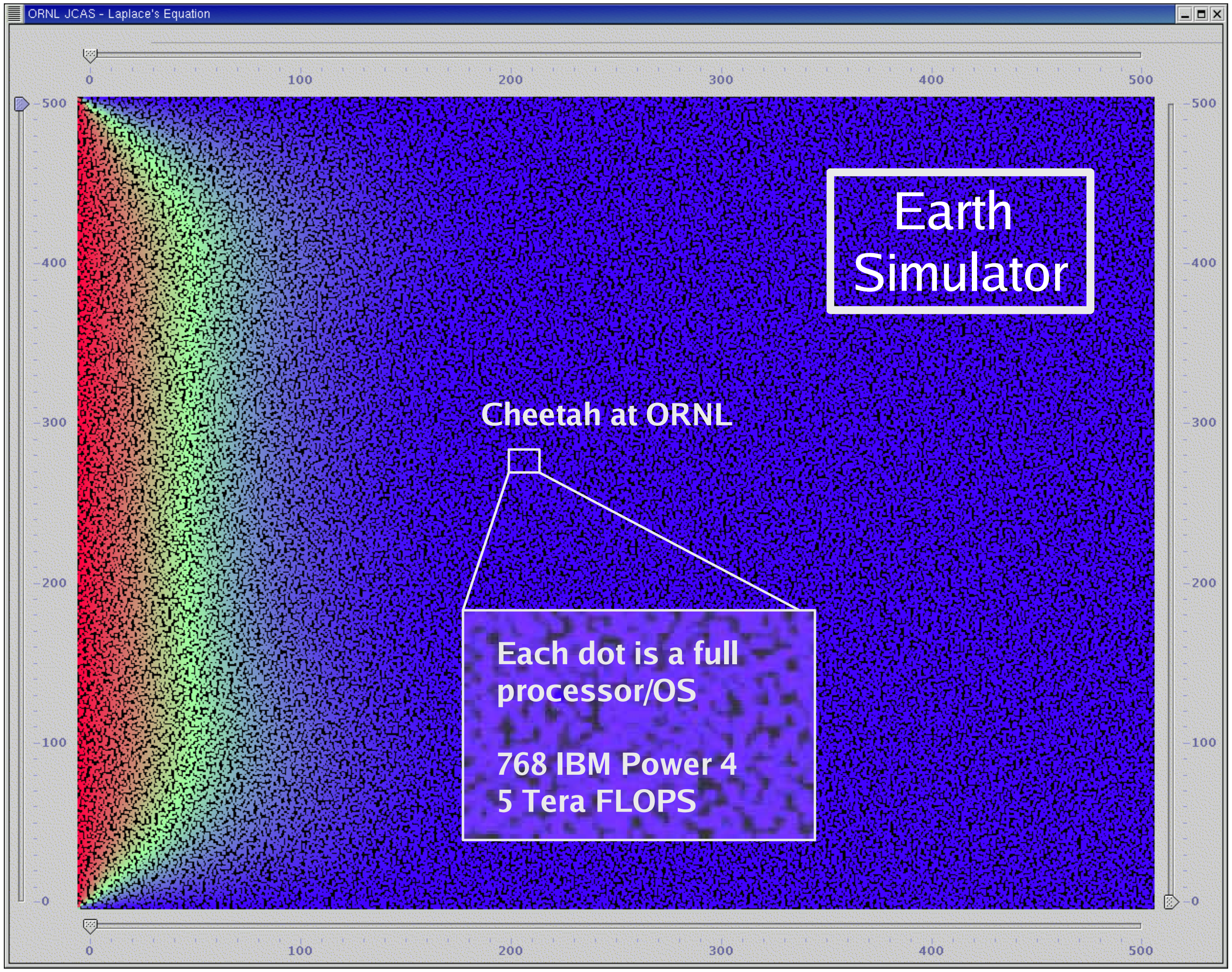
ORNL/IBM Collaboration

- Development of biology and material science applications for Blue Gene/L.
- Theory of super-scalar algorithms.
  - Natural fault-tolerance.
  - Scale invariance.
- Focus on test and demonstration tool.
- *Get scientists to think about fault-tolerance in super-scale systems.*

Java Cellular Architecture Simulator - JCAS

- Runs as distributed application.
- Simulates up to 1,000,000 threads.
- Standard network configurations:
  - Multi-dimensional mesh.
  - Multi-dimensional torus.
- Experimental network configurations:
  - Grid positions, nearest/random neighbors.
  - Random positions, nearest/random neighbors.
- Simulation is not in real-time.

Java Cellular Architecture Simulator - JCAS



Super-scale Fault-tolerance

- Does it makes sense to restart all 65,535 processors because one failed?
- The mean time between failures is likely to be 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

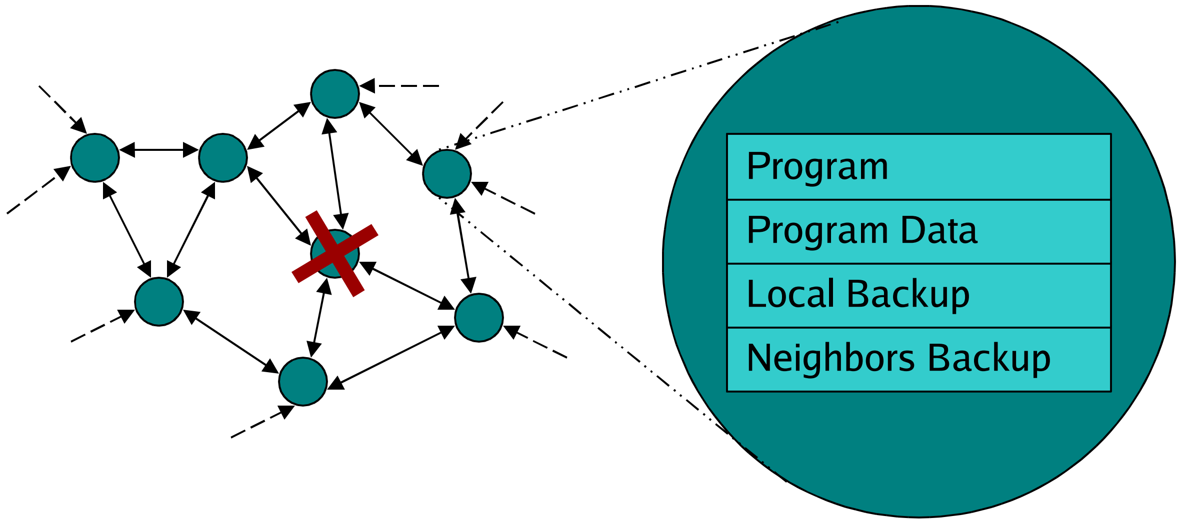
- Coordinated backup to the memory of dedicated checkpoint processors.
- Reduces overhead and latency.
- Allows more frequent checkpoints.
- Shorter application running time.
- In case of a failure:
  - Rollback to local memory backup.
  - Restart from remote memory backup.

Diskless Checkpointing

- A checkpoint processor may become a replacement processor during recovery.
- New checkpoint processors can be dynamically added during runtime.
- Encoding semantics (RAID) trade off storage size vs. degree of fault tolerance.
- Infrequent checkpointing to stable storage (disk/tape) of backup.

Diskless Checkpointing on Super-scale Architectures

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



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 based on system communication infrastructure.

Backup Coordination

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

Global Synchronization

- Global application snapshot.
- Synchronous backup of all local states.
- Global barrier at stable application state.
- Synchronizes complete application.
- Easy to implement.
- Preferred method for communication intensive applications.

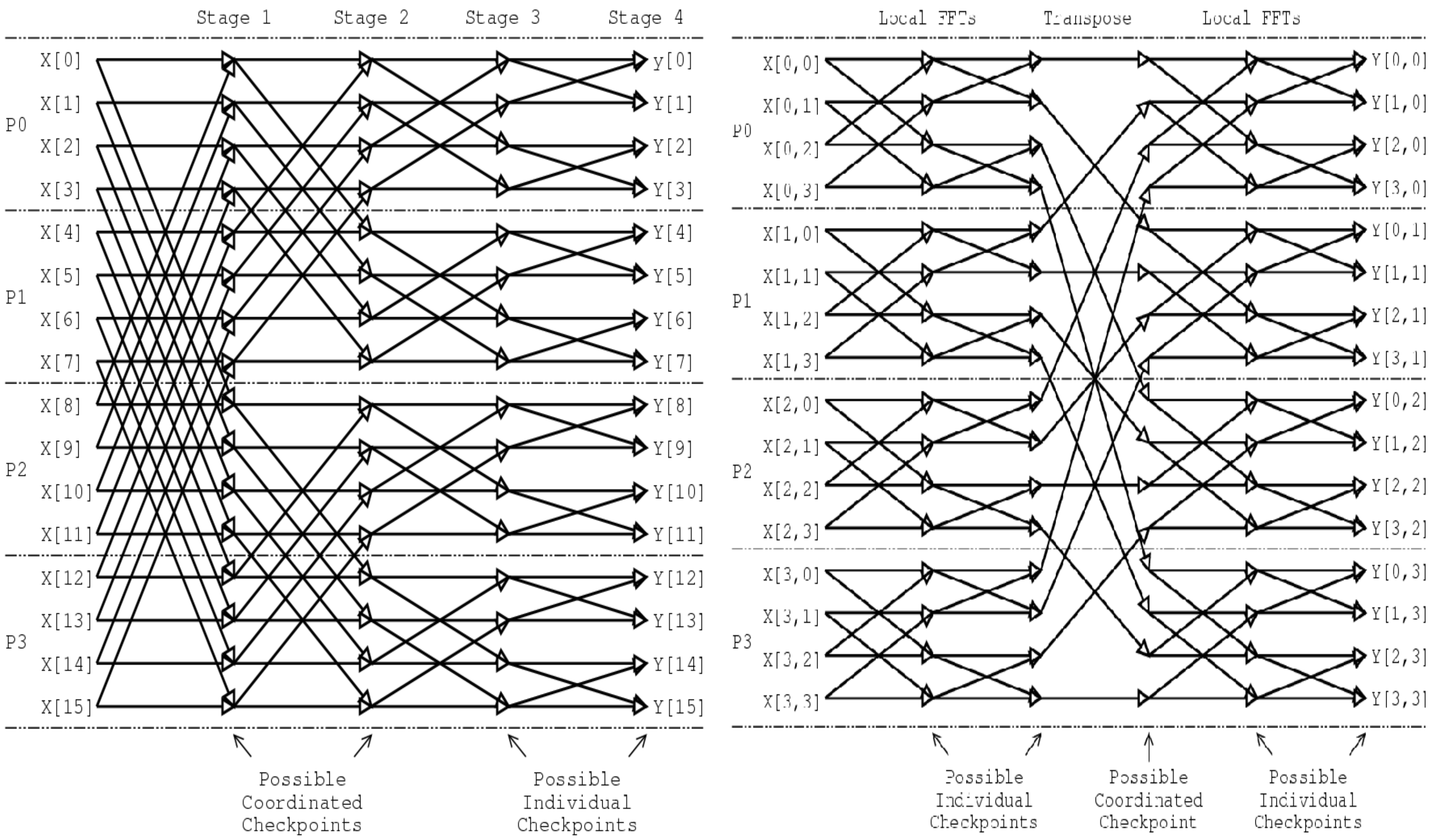
Local Synchronization

- Asynchronous individual backup of local state and in-flight messages.
- Acknowledgements for messages to keep accurate records of in-flight messages.
- Additional local group communication.
- Different methods to retrieve missed messages from neighbors.
- More complicated to implement.
- Preferred method for less communication intensive applications.

Application to the FFT

- Distributed and transposed FFT:
  - Not naturally fault-tolerant.
  - Not scale invariant.
  - Mixture of local and global communication.
  - Ideal test scenario for diskless checkpointing.
- Other Fourier transform algorithms may be naturally fault-tolerant or scale better.
- They are not considered here to test the super-scale diskless checkpointing.

How to checkpoint FFT?



Observations

- Simulation on up to 100,000 threads.
- Global synchronization proved to be easier to implement and performs better for the communication intensive FFT.
- Local synchronization was complicated to realize, and may perform better for other algorithms with less communication.
- Timing, latency and bandwidth data impossible to obtain from this simulation.

Conclusions

- Diskless peer-to-peer checkpointing on super-scale architectures is possible.
- Tests showed strengths and weaknesses of different synchronization methods.
- Currently only tested on JCAS simulator.
- Real-time tests with different applications are needed for further discussion.
- Final real-world implementation requires super-scalable FT-MPI or PVM.
- *A lot of work still needs to be done.*

Diskless Checkpointing on Super-scale Architectures

Applied to the Fast Fourier Transform

Christian Engelmann, Al Geist
Network and Cluster Computing Group
Computer Science and Mathematics Division
Oak Ridge National Laboratory, Oak Ridge, USA