On the Role of Deterministic Fine-Grain Data Synchronization for Scientific Applications: A Revisit in the Emerging Many-Core Era

Weirong Zhu , Ziang Hu, Guang R. Gao
Department of Electrical & Computer Engineering
University of Delaware
2007-03-30

Outline

- Introduction
- Experience of Parallelizing scientific kernels on a many-core architecture
- Experimental results
- Conclusion

Many-Core Era

- Multi-core (2-8 cores) begins to dominate the market.
- Many-core (10s or 100+ cores) is emerging. [Intel Platform 2015, Bill Dally's Kenote on ICCD'06 , Steve Pawlowski's Keynote on HPCA'07]
- Examples:
  - Intel 80-core TeraScale chip & Larrabee chip
  - IBM Cyclops-64 chip with 160 thread units
  - ClearSpeed 96-core CSX chip
  - Cisco 188-core Metro chip

Fine-Grain Synchronization

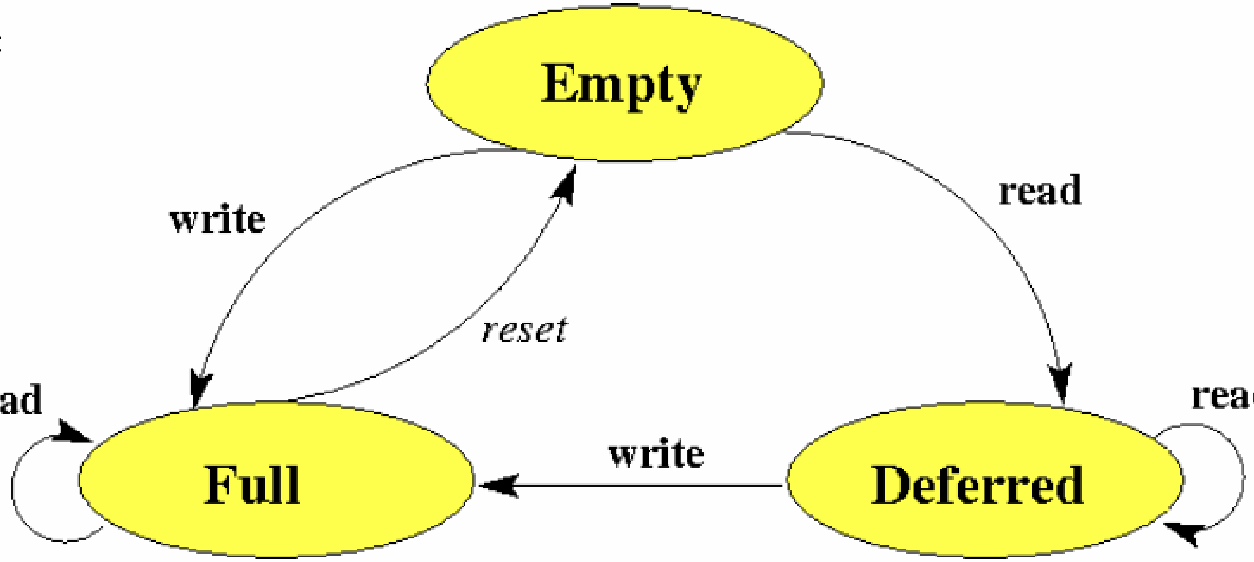
- Many-core: massive intra-chip parallelism
- Shared memory programming model
- Multithreading: maintain a large number of active threads to exploit the parallelism
- Efficient fine-grain synchronization determines the granularity of parallelism can be exploited. [ChenEtAl1990, TullsenEtAl1999]

Synchronization

- Under shared-memory programming model, synchronization enforces
  - mutual exclusion
  - **read-after-write data dependency**
- Term used in this presentation
  - *Fine-grain data synchronization*: enforce RAW data dependency on word-level between threads

An Example of Fine-Grain Data Synchronization: I-structure

I-structure



- Fine-grain synchronization: Associate "state" to a memory location (fine granularity). Synchronization for the memory location is realized through "state transition" on such state.
- Interface to software: perform read/write as well as synchronization directly on the memory location.

Scientific Applications

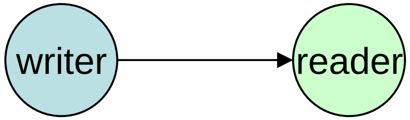
- An important class of the target applications for many-core
- Intrinsically deterministic
- When running in parallel with multithreading
  - data dependencies in the code should be enforced effectively & efficiently

Fine-grain Data Synchronization: hardware support

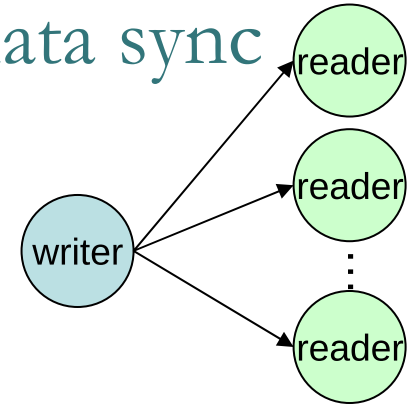
- On previously built multiprocessors:
  - Full/empty bits on Tera, MTA-2, etc.
  - I-structure and M-structure on dataflow model based machines.
- On many-core architectures:
  - In this paper, we assume the use of *SSB: Synchronization State Buffer* [ISCA2007]

SSB: Synchronization State Buffer

- SSB: A small buffer attached to the memory controller of each memory bank
- Record and manage the states of *active synchronized data units* [ISCA2007]
- Support word-level fine-grain data sync
- We will use



Single-writer-single-reader



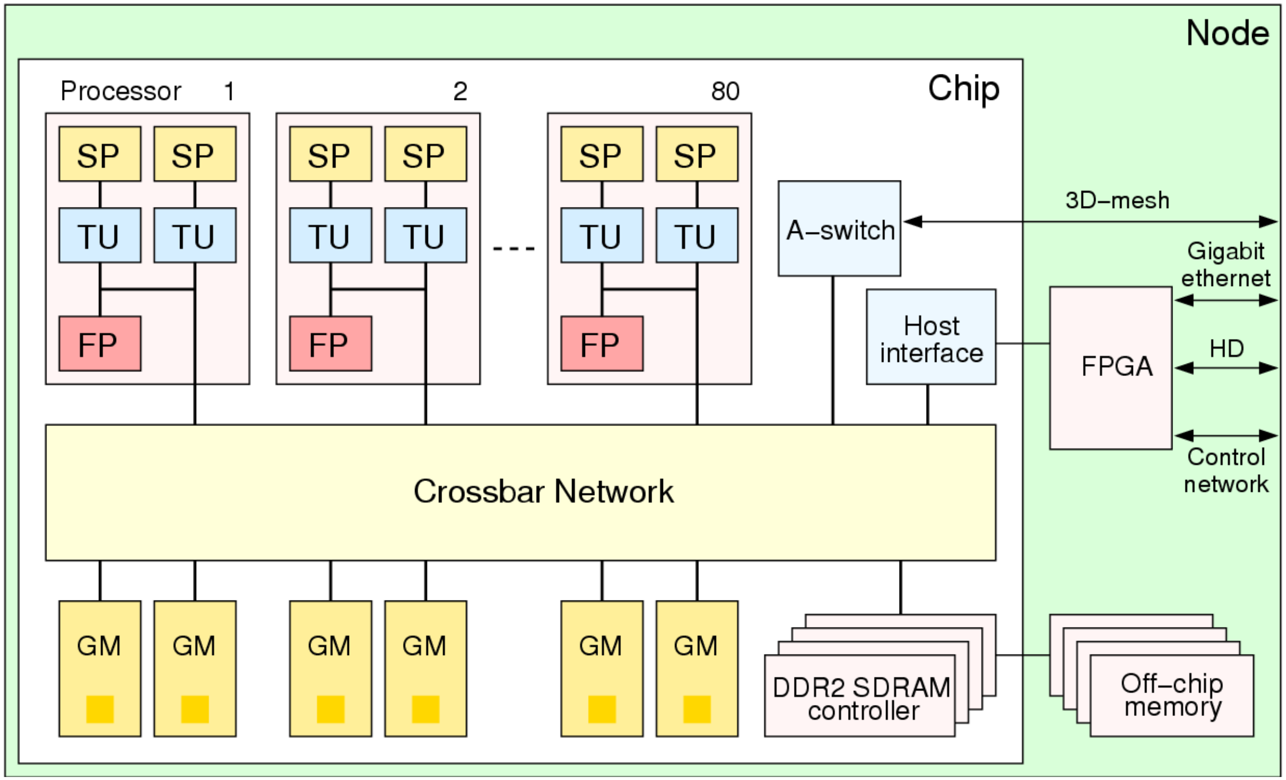
Single-writer-multiple-reader

Parallelization of Scientific Kernels

- We report our experience on parallelizing three representative scientific kernels on a many-core architecture
  - 1D Laplace solver: iterative computation
  - Liner recurrence equations: irregular pattern of data dependencies
  - Wavefront computation: wavefront-like propagated computation
- Key: how fine-grain data sync. is used.

Experimental Infrastructure

IBM Cyclops-64 Chip Architecture

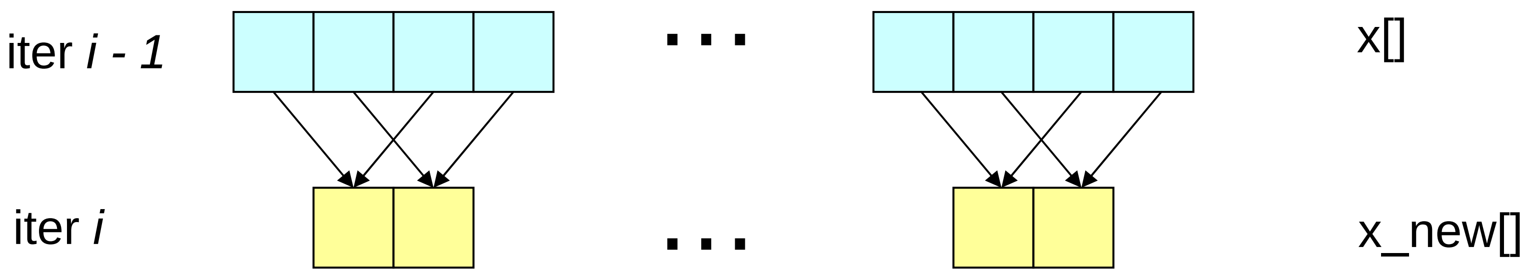


- 160 hardware thread units
- three-level explicit memory hierarchy
- efficient thread-level execution support
- SSB is implemented in simulator: 16-entry, 8-way associative per SRAM bank

1D Laplace Solver

- A finite difference method to achieve numerical approximation of Laplace's equation

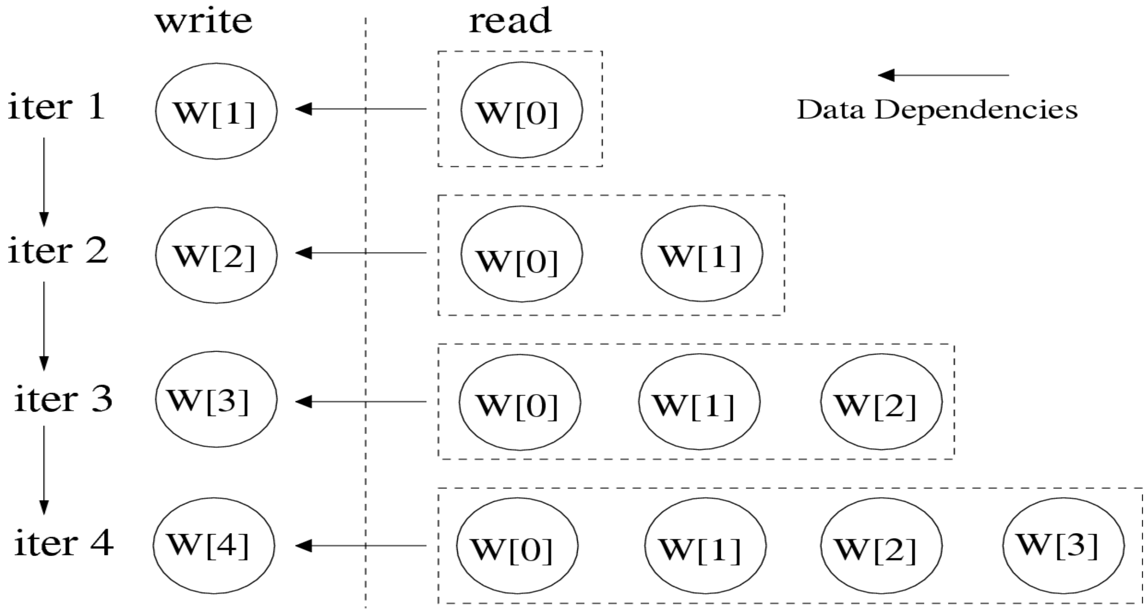
$$x\_new[i] = 0.5 \times \left(x[i-1] + x[i+1] + b[i] \right)$$



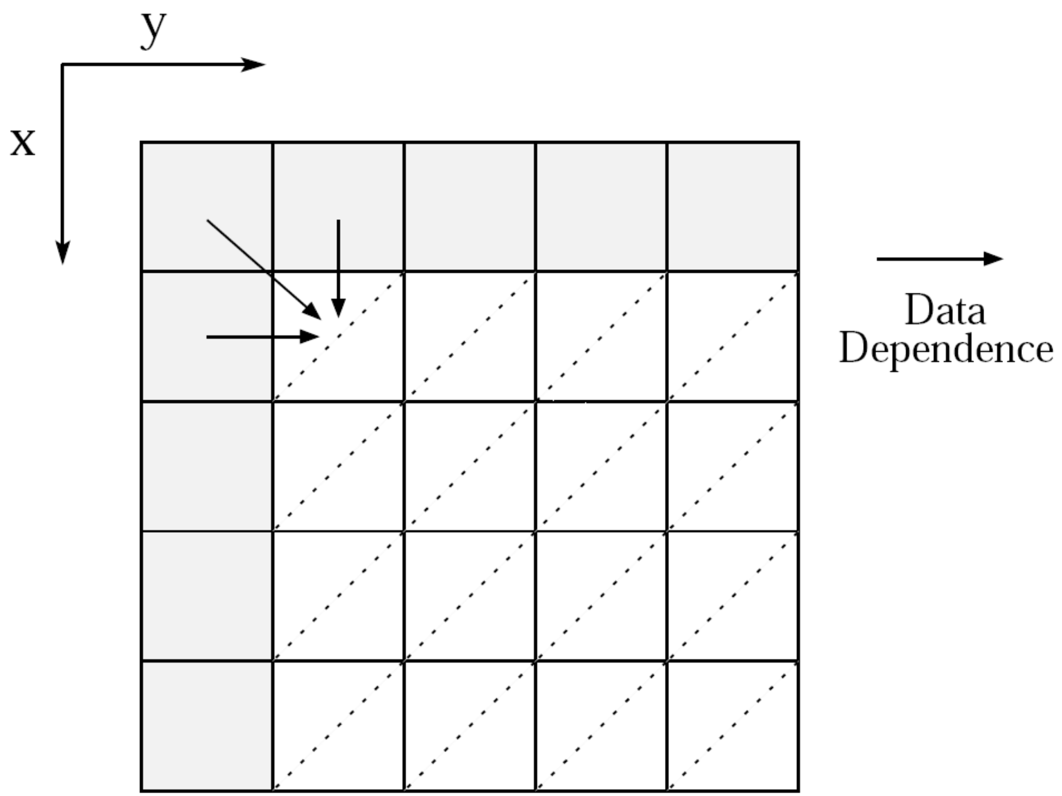
Linear Recurrence Equation

- Livermore Loop 6

```
for ( i=1 ; i<n ; i++ )
    for ( k=0 ; k<i ; k++ )
        W[i] += b[k][i] * W[(i-k)-1];
```



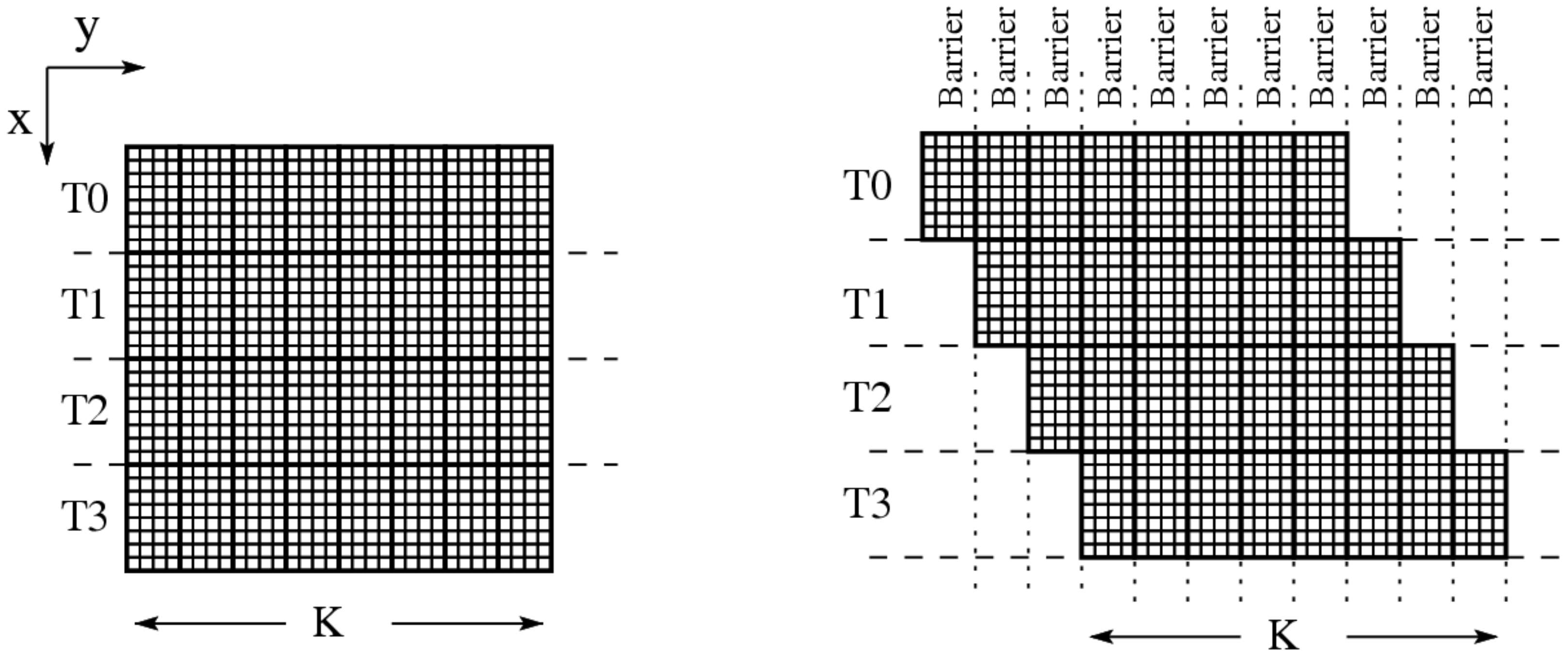
Wavefront Computation



Parallelization of Kernels

- Coarse-Grain version: using barrier (implemented using C64 on-chip signal bus directly)
  - all-to-all, all consumers waits for all producers
- Fine-Grain version: using fine-grain data synchronization supported by SSB.
  - point-to-point: using word-level synchronized read/write to enforce RAW data dependencies inherent in the application.

Wavefront Computation: Coarse-Grain Parallelization

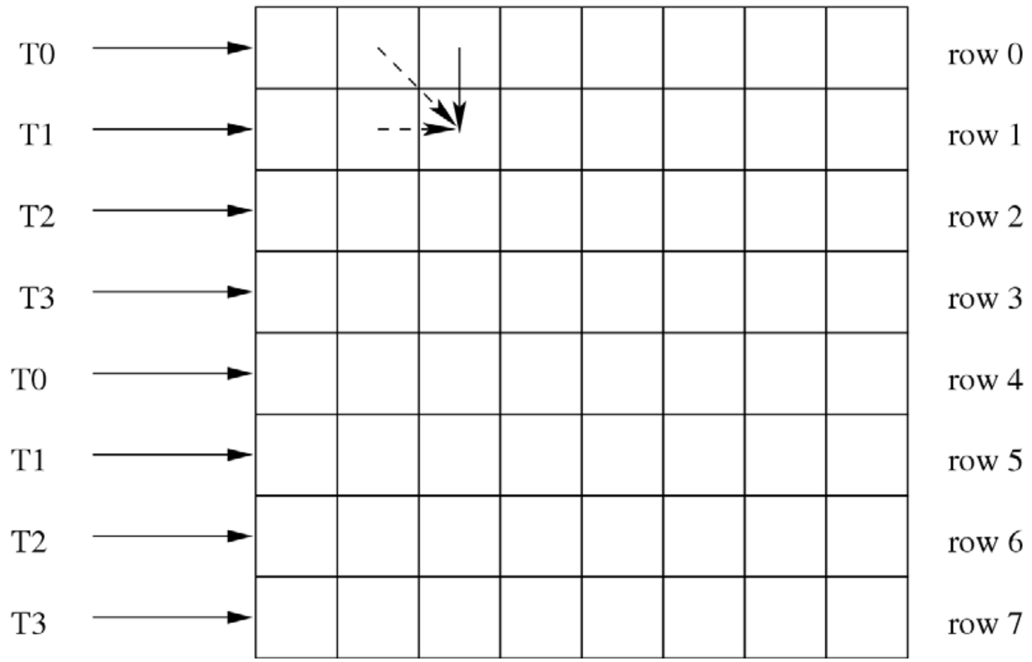


(a) Partition the Solution Space

(b) Computation Scheme in Parallel

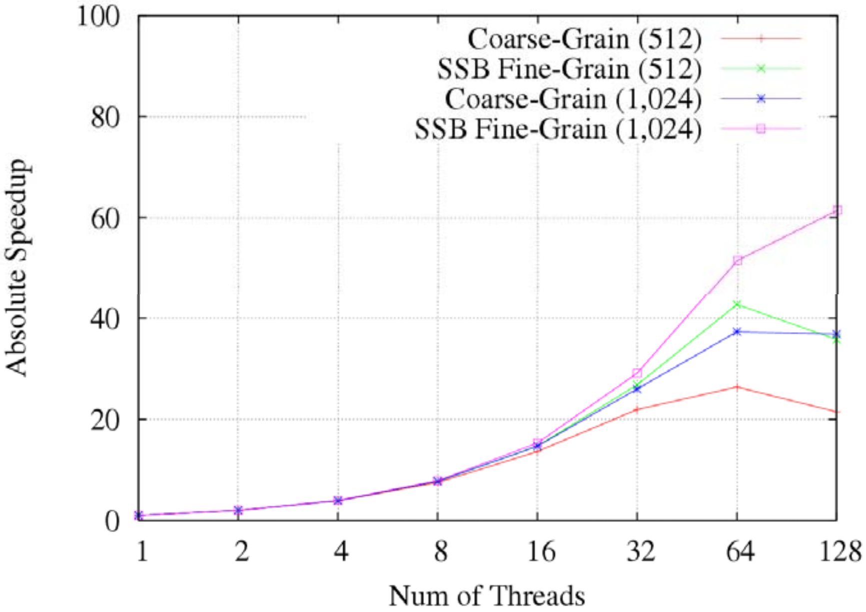
K: determines the granularity of data associated with each barrier

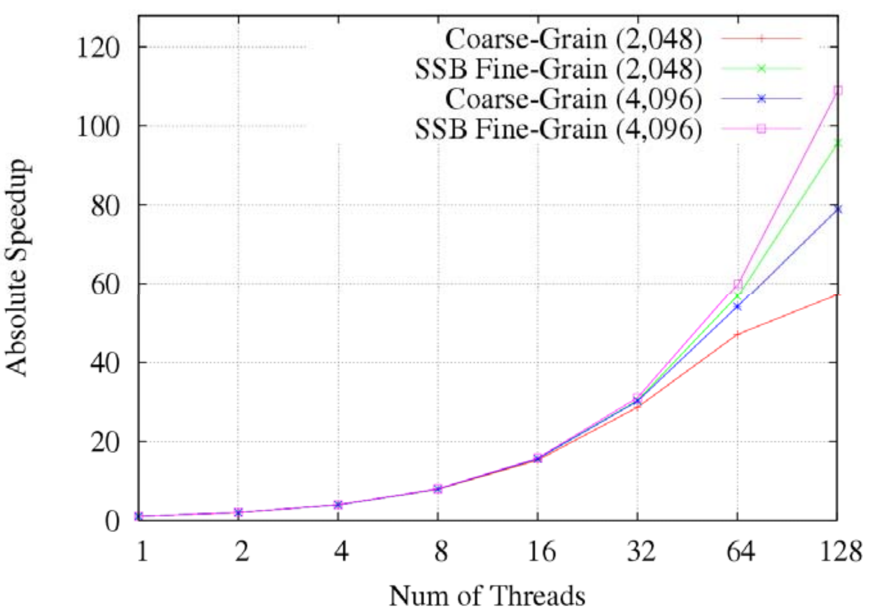
Wavefront Computation: Fine-Grain Parallelization



- Use fine-grain data sync to enforce the data dependencies
- No barrier

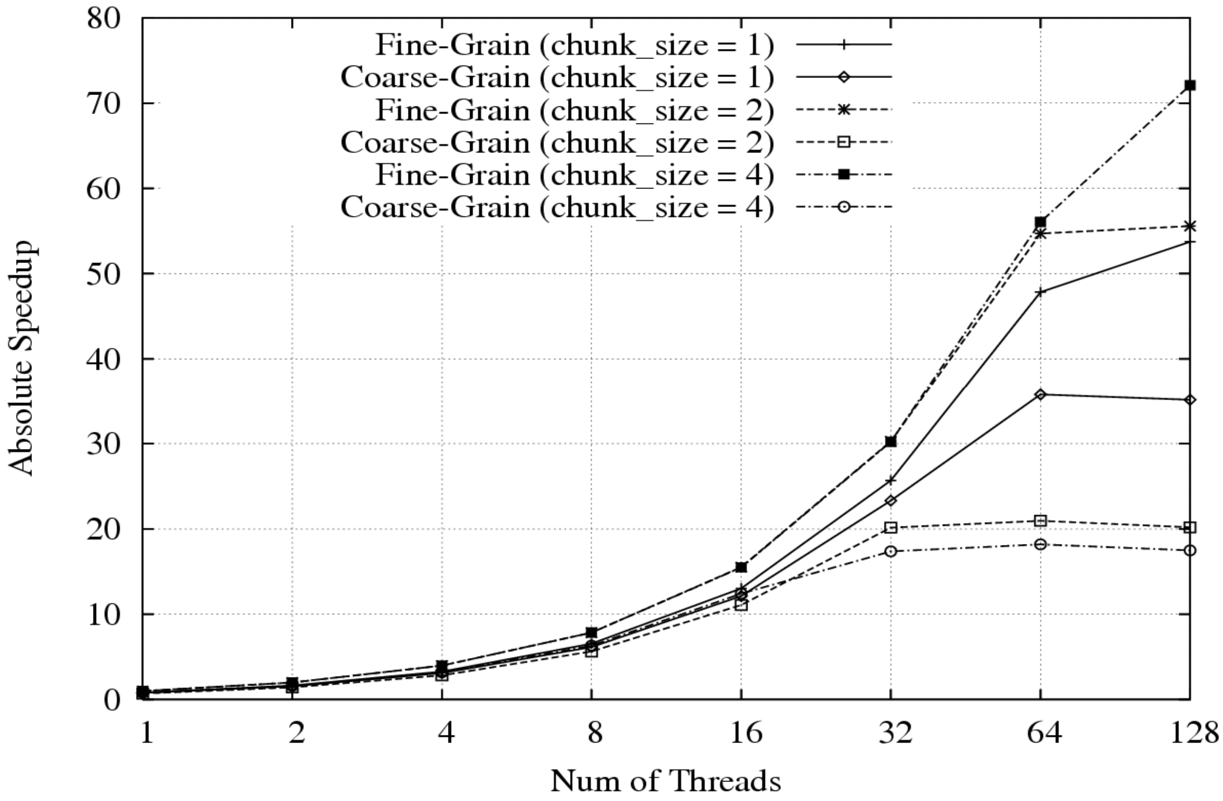
1D Laplace Solver





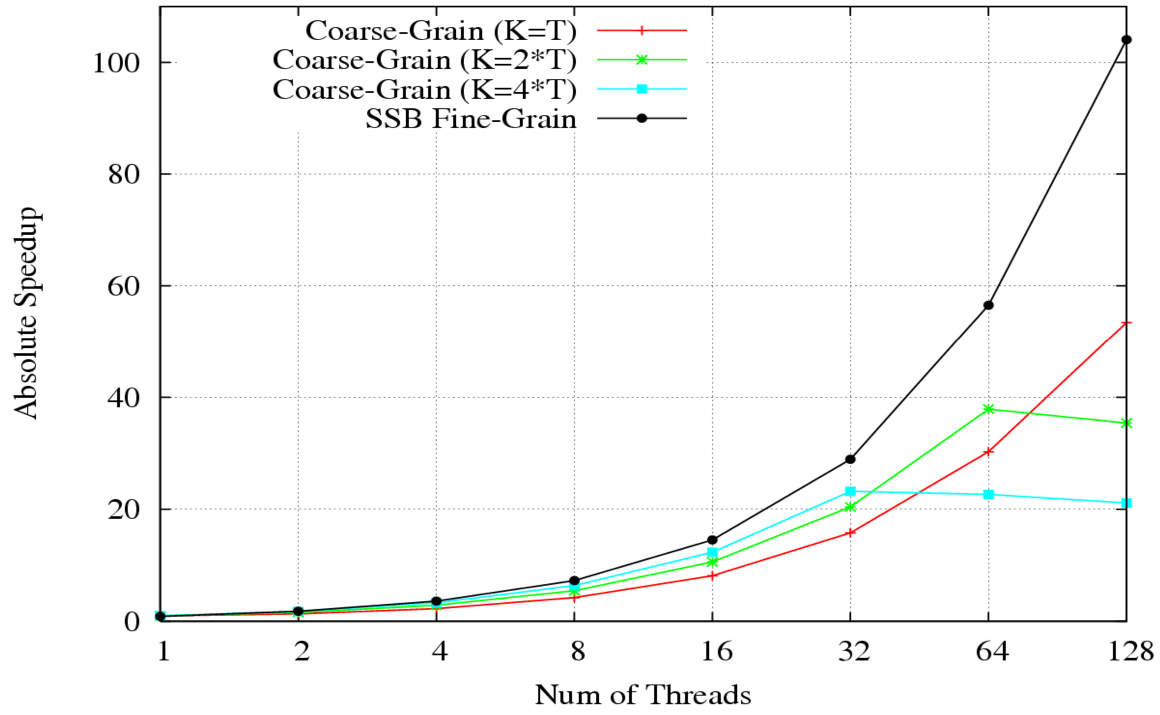
When runs on 128 threads with a problem size of 4,096, the fine-grain version achieves a speedup of 109, outperforms the coarse-grain one by 38.1%

Linear Recurrence Equations



When runs on 128 threads, with chunk_size = 4, fine-grain version outperforms the coarse-grain version by 312%

Wavefront Computation



When runs on 128 threads, the fine-grain version achieves a speedup of 104, outperforms the three coarse-grain versions by 94.9%, 192.4%, and 392.7% respectively.

Conclusion

- We reported our experience on parallelization of three representative scientific application kernels on a many-core architecture: 160-core C64
- We showed that fine-grain sync. can be used to enforce RAW data dependency among threads, and avoid unnecessary waiting and global communication due to the use of coarse-grain barrier
- Significant performance benefit can be observed when number of threads are large

Acknowledgement

- Monty Denneau, architect of IBM C64
- Support from IBM, ETI, DoD, DoE (DE-FC02-01ER25503), NSF (CNS-0409332), and other government sponsors
- CAPSL group at University of Delaware
- Special thanks to Vugranam C. Sreedhar (IBM), Ioannis Venetis, and Juan del Cuvillo
- Thanks for feedback from anonymous reviewers

Thank You