High Performance Simulations of Electrochemical Models on the Cell Broadband Engine

James Geraci

HPEC Workshop

September 20, 2007

This work is sponsored by the Department of the Air Force under Air Force contract FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government

Outline

- **Introduction**
- **Out of Core Algorithm**
- **In Core Algorithm**
- **Summary**

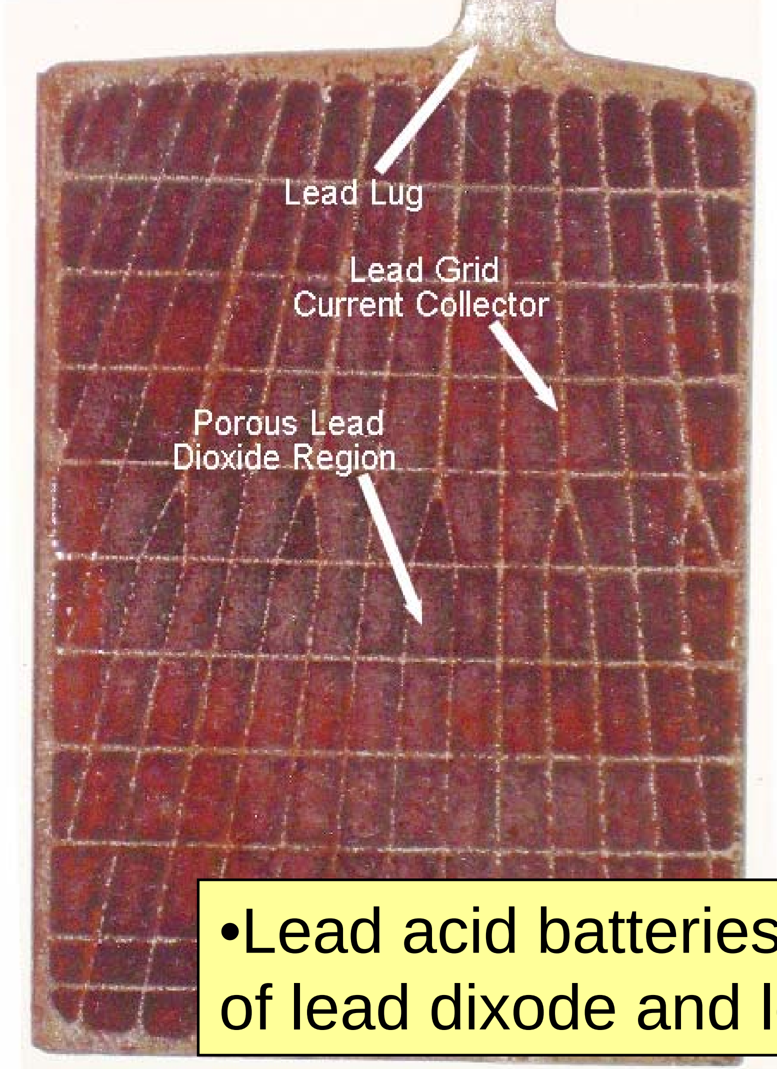
- ***Introduction***
- ***Modeling battery physics***
- ***LU decomposition***
- ***CELL Broadband Engine***

Introduction
Real Time Battery State of Health Estimation

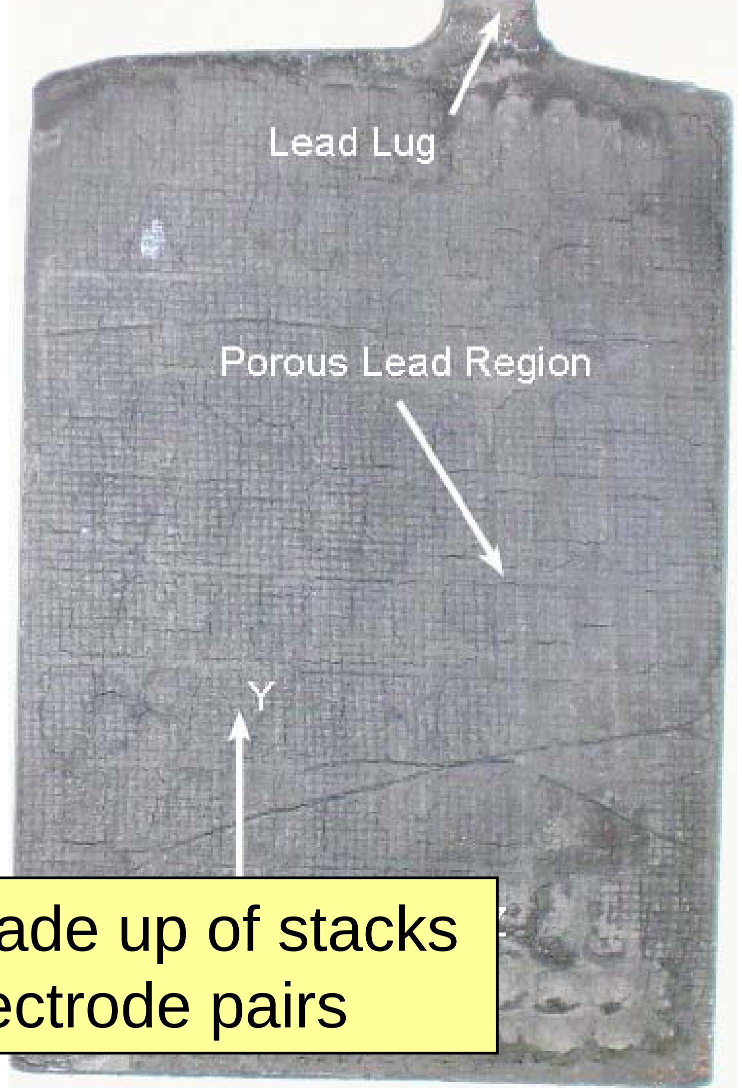
- The way in which a battery ages is a strong function of battery geometry.
- The rate of self discharge is also a function of battery geometry.

Inside a Lead Acid Battery

Face View Lead Dioxide Plate



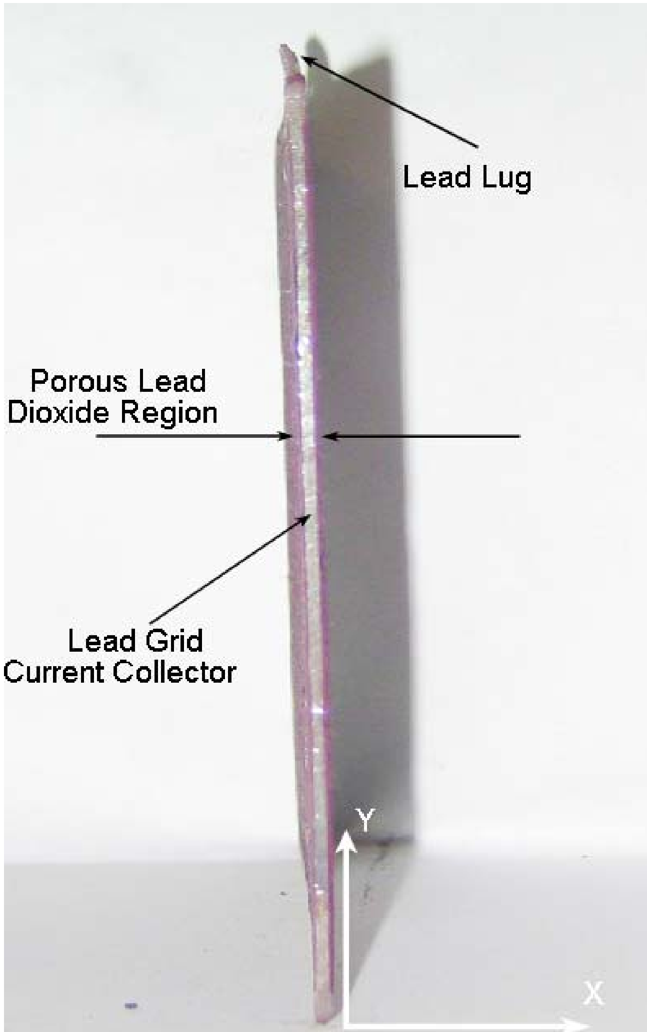
Face View Lead Plate

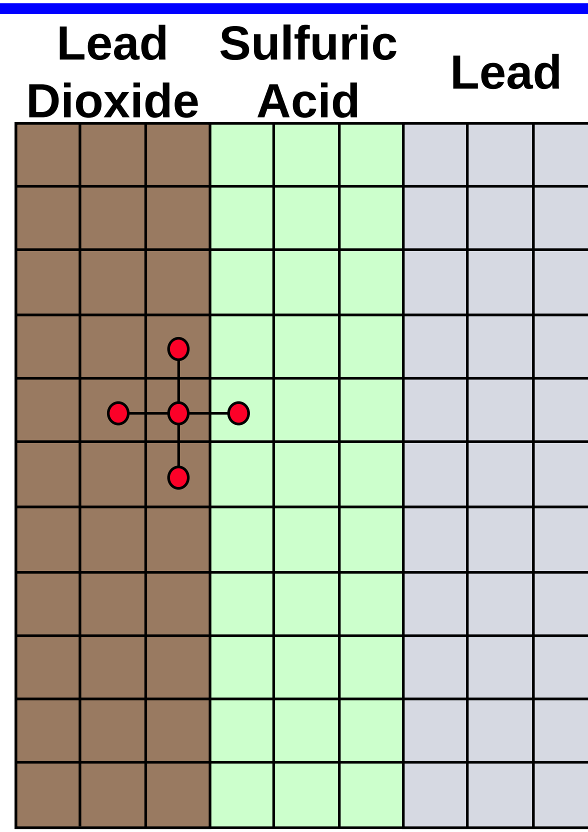


•Lead acid batteries are made up of stacks of lead dixode and lead electrode pairs

Finite volume description of battery

Edge view lead dioxide plate

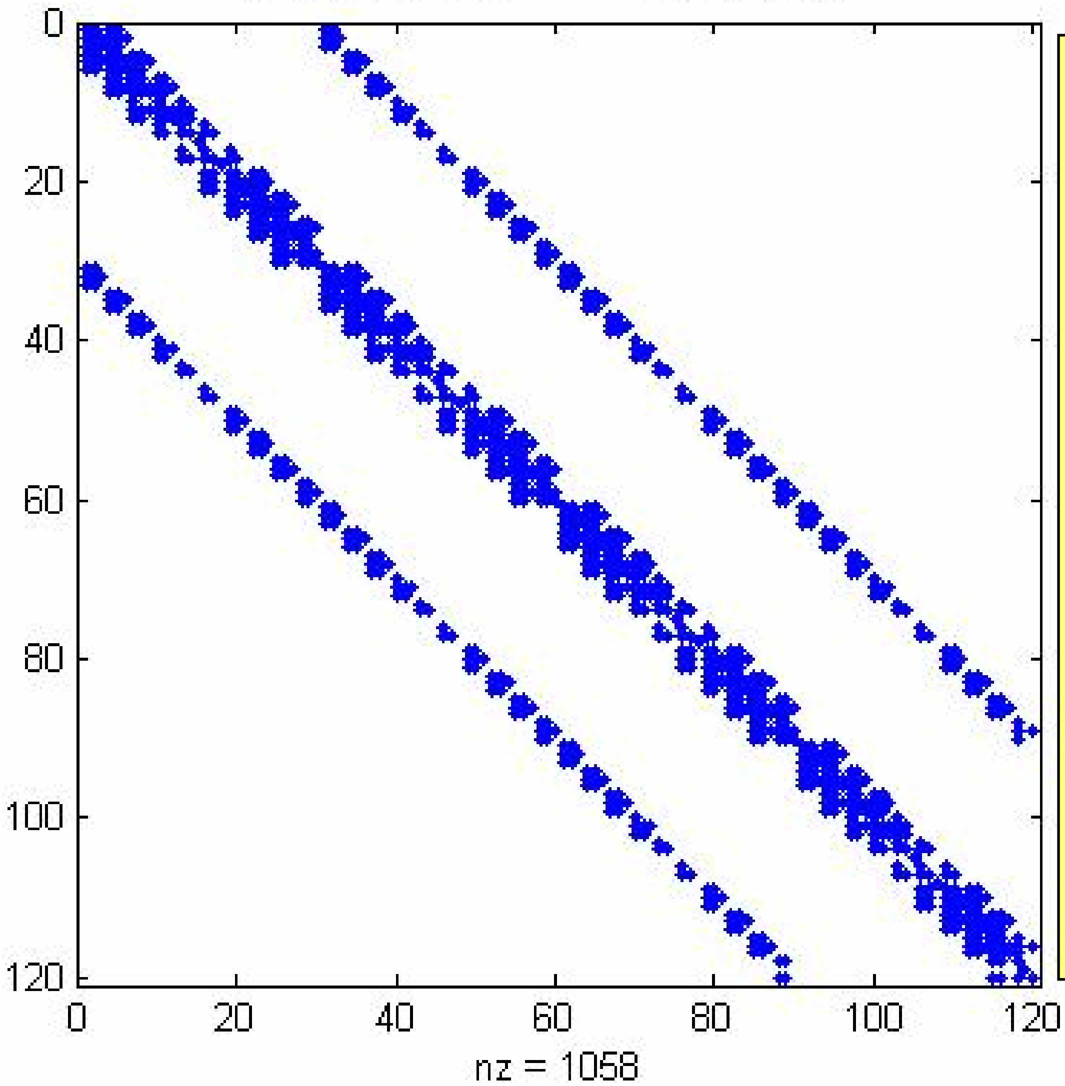




- The center volume's physics can be influenced by the physics of the volumes to the right, the left, above, and below

- The 5 point stencil gives rise to banded matrix

Matlab spy plot of Banded Matrix



- Non-zero entries shown in blue
- Inner band around diagonal
- Distant narrow outer band
- For the physics of a lead acid battery, the matrix is not symmetric (shown)
- For the physics of a lead acid battery, the matrix is poorly conditioned 10^8-10^12
  - need double precision, no GPGPU

Outline

- **Introduction**
- **Out of Core Algorithm**
- **In Core Algorithm**
- **Summary**

- ***Introduction***
- ***Modeling battery physics***
- ***LU decomposition***
- ***CELL Broadband Engine***

LU Decomposition

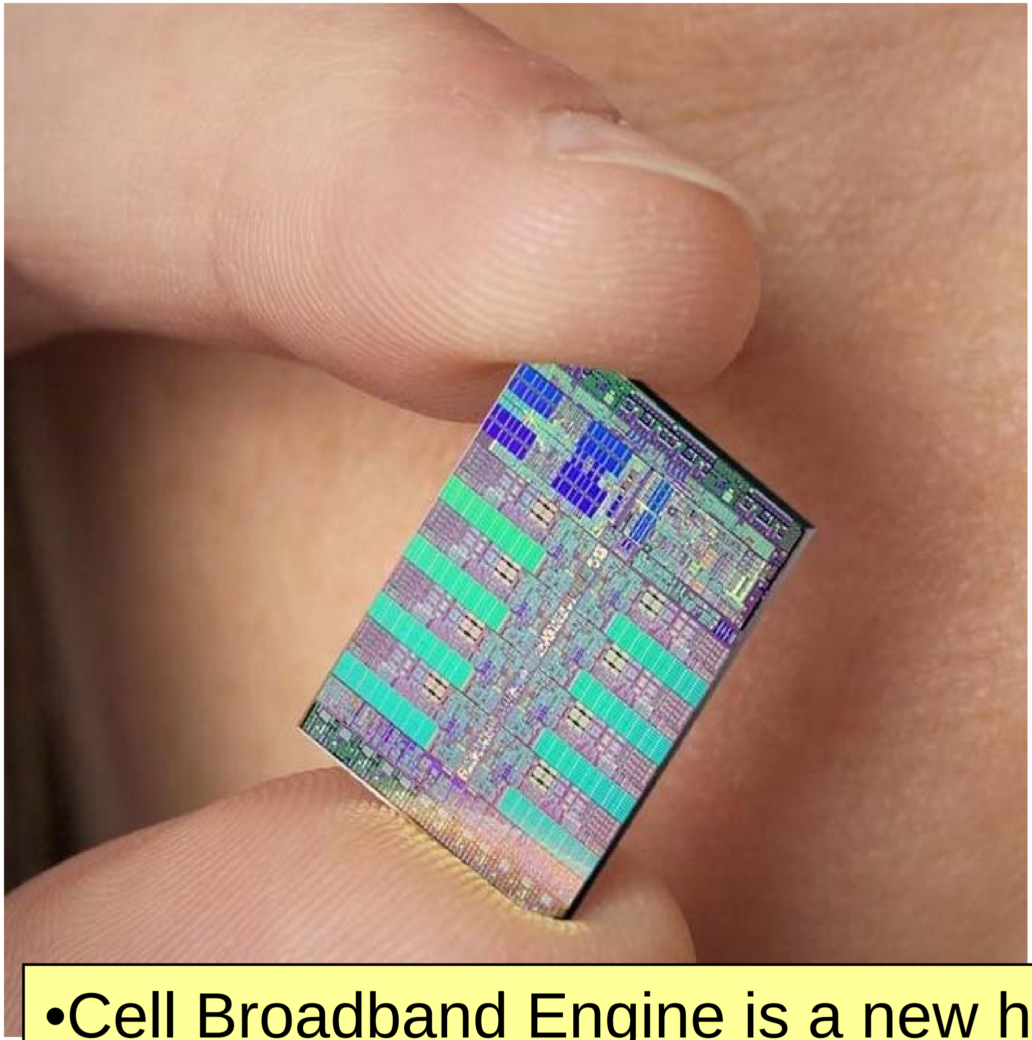
- LU decomposition decomposes a matrix J into a lower triangular matrix L and an upper triangular matrix U
- L & U can be used to solve a system of linear equations Jx = r by forward elimination back substitution
  - Essentially Gaussian Elimination
- Often used on poorly conditioned systems where ‘iterative solvers’ can’t be used.
- Difficult to parallelize for small systems because of the fine grain nature of the parallelism involved.
- Banded LU is a special case of LU
  - The matrix J has a special ‘banded’ data pattern.

Outline

- **Introduction**
- Out of Core Algorithm
- In Core Algorithm
- Summary

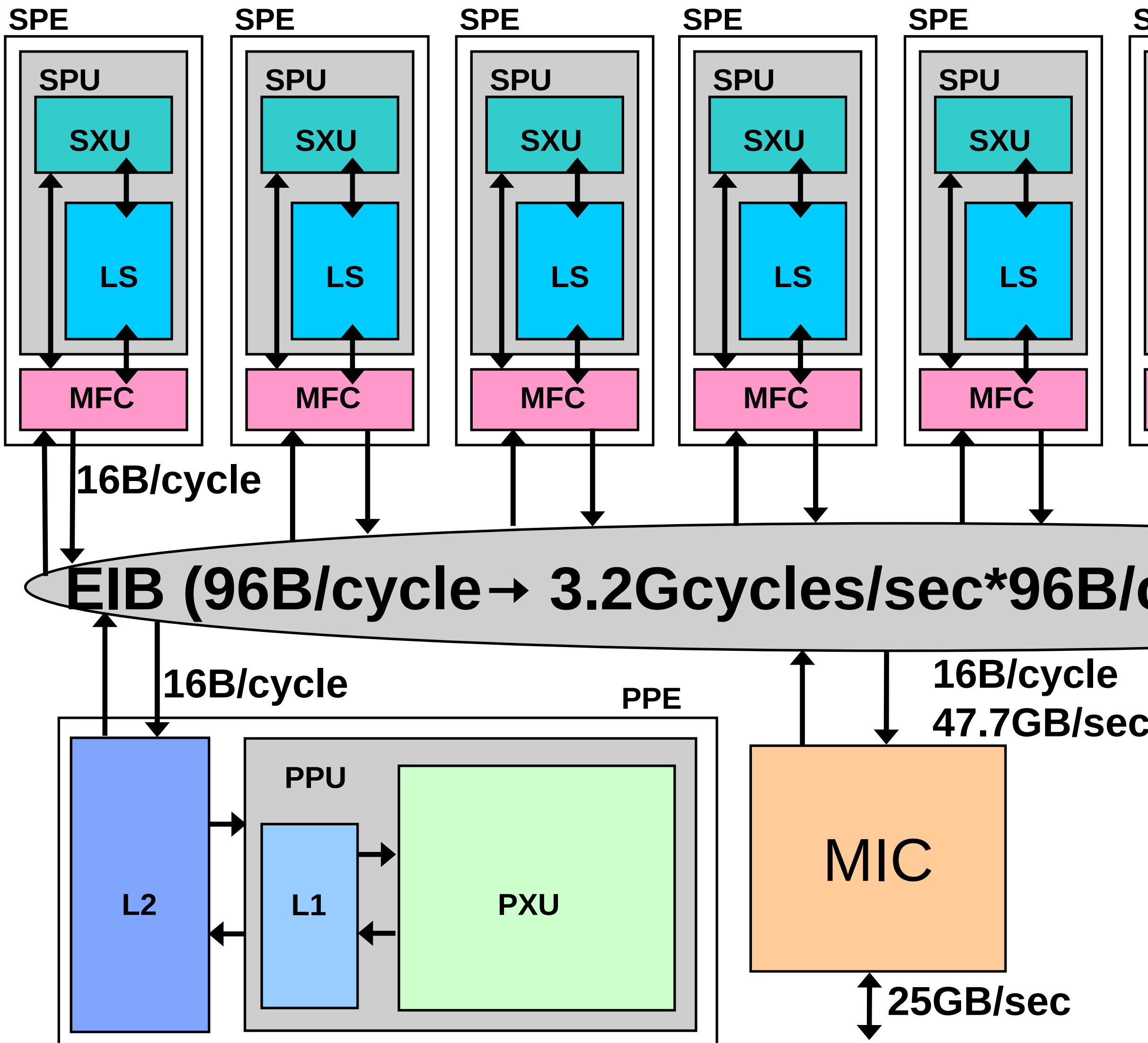
- *Introduction*
- *Modeling battery physics*
- *LU decomposition*
- ***CELL Broadband Engine***

Cell Broadband Engine



•Cell Broadband Engine is a new heterogeneous multicore processor that features large internal and off chip bandwidth.

Cell Broadband Engine



Bandwidth:
- 50GB/sec/SPU
- 25GB/sec off chip

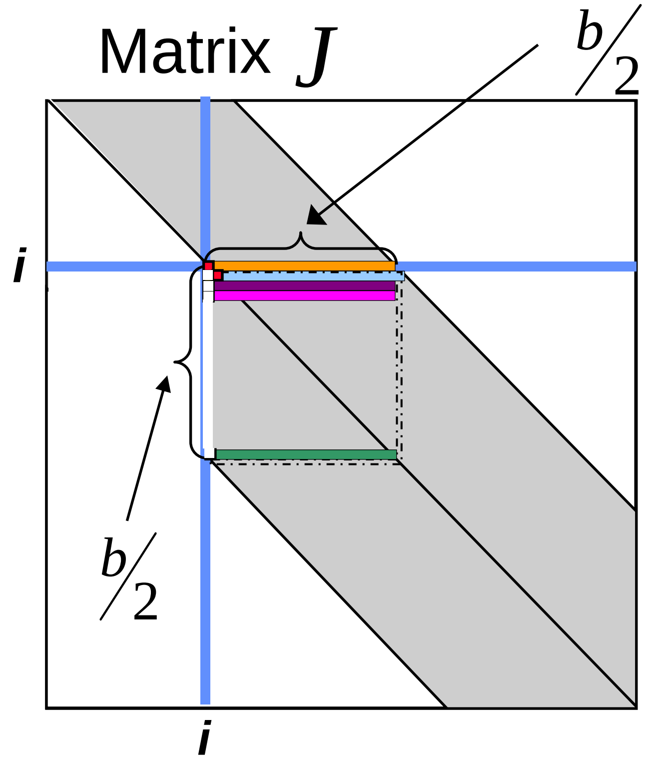
SPE Performance:
- 14 Gflops double

Outline

- Introduction
- **Out of Core Algorithm**
- In Core Algorithm
- Summary

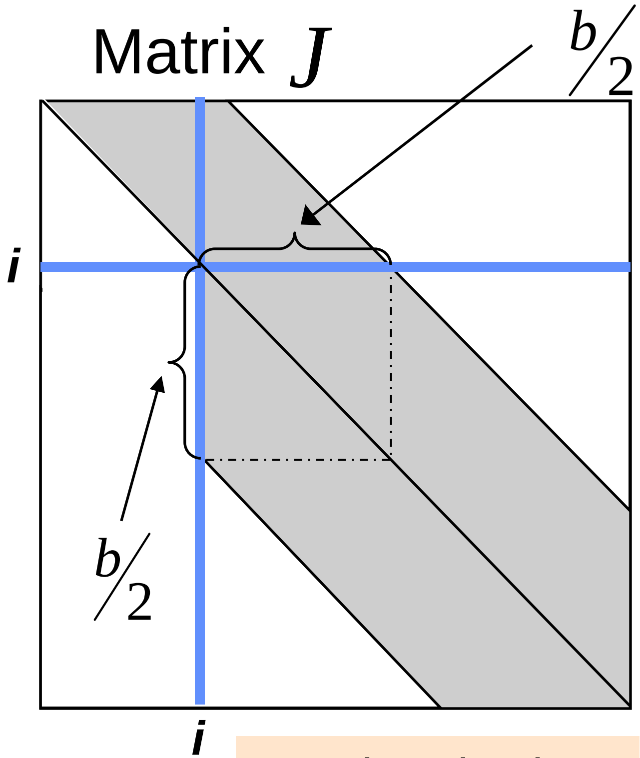
- ***Banded LU***
- *Performance*
- *Latency*
- *Synchronization*

Banded LU
Out of Core Algorithm Explained



$$J(i+1:i+b/2,\, i:i+b/2) -= \frac{J(i+1:i+b/2,\, i)}{J(i,i)} J(i,\, i:i+b/2)$$

Banded LU
Out of Core Algorithm Analyzed



	Compute		$2N\left(b/2\right)^2$
$\forall i < N$	**Multiples:**		$\left(b/2\right)^2$
	Subtracts:		$\left(b/2\right)^2$
	Memory Accesses		$2N\left(b/2\right)$
	Memory Doubles Moved		$2Nb\left(b/2\right)$
$\forall i < N$	**Gets:**	$b/2$	b
	Puts:	$b/2$	b

Synchronizations: N

- Compute/Memory Ratio = ½
- For a 16728x16728 matrix with band size of 420, almost 22 GB of data would have to be moved.

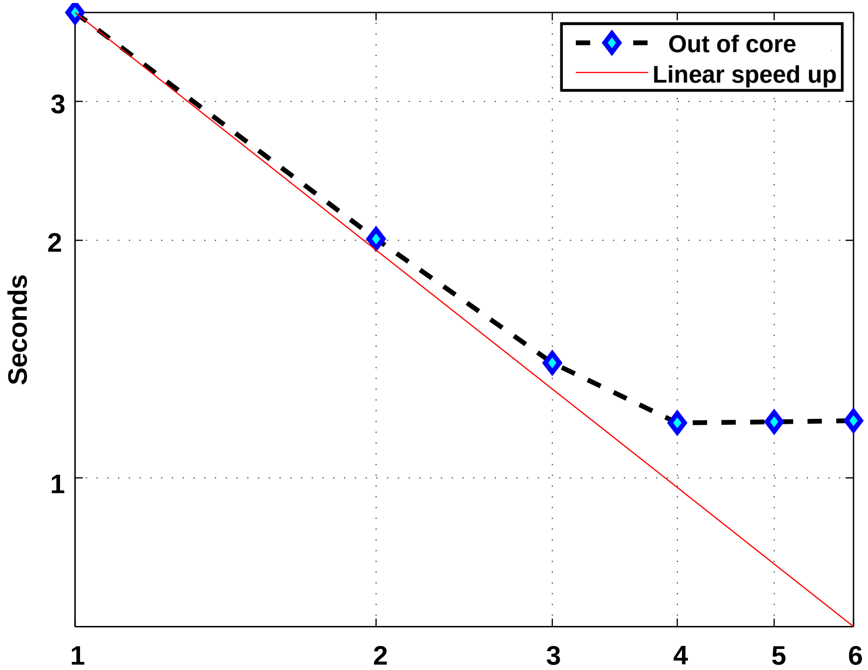
Outline

- Introduction
- **Out of Core Algorithm**
- In Core Algorithm
- Summary

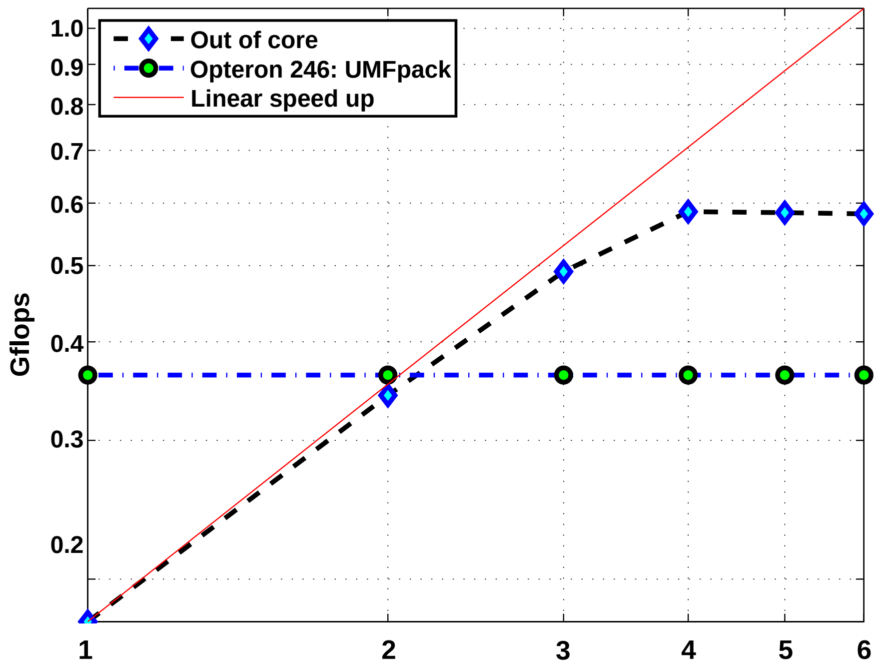
- *Banded LU*
- ***Performance***
- *Latency*
- *Synchronization*

Peformance of Out of Core Algorithm

Compute Time for matrix size 16728x16728 w/ band size of 420



Gflops for matrix size 16728x16728 w/ band size of 420



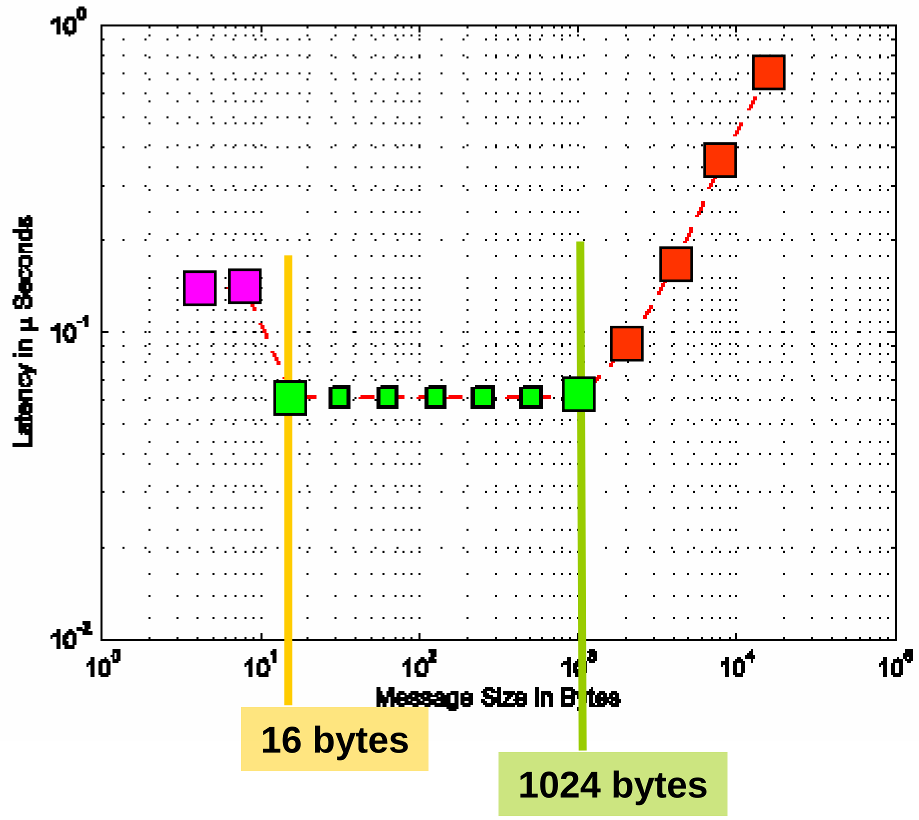
- Out of Core Algorithm outperforms UMFpack on Opteron 246 based workstation.
- No appreciable gain in performance past 4 SPEs

Outline

- Introduction
- **Out of Core Algorithm**
- In Core Algorithm
- Summary

- *Banded LU*
- *Performance*
- ***Latency***
- ***Synchronization***

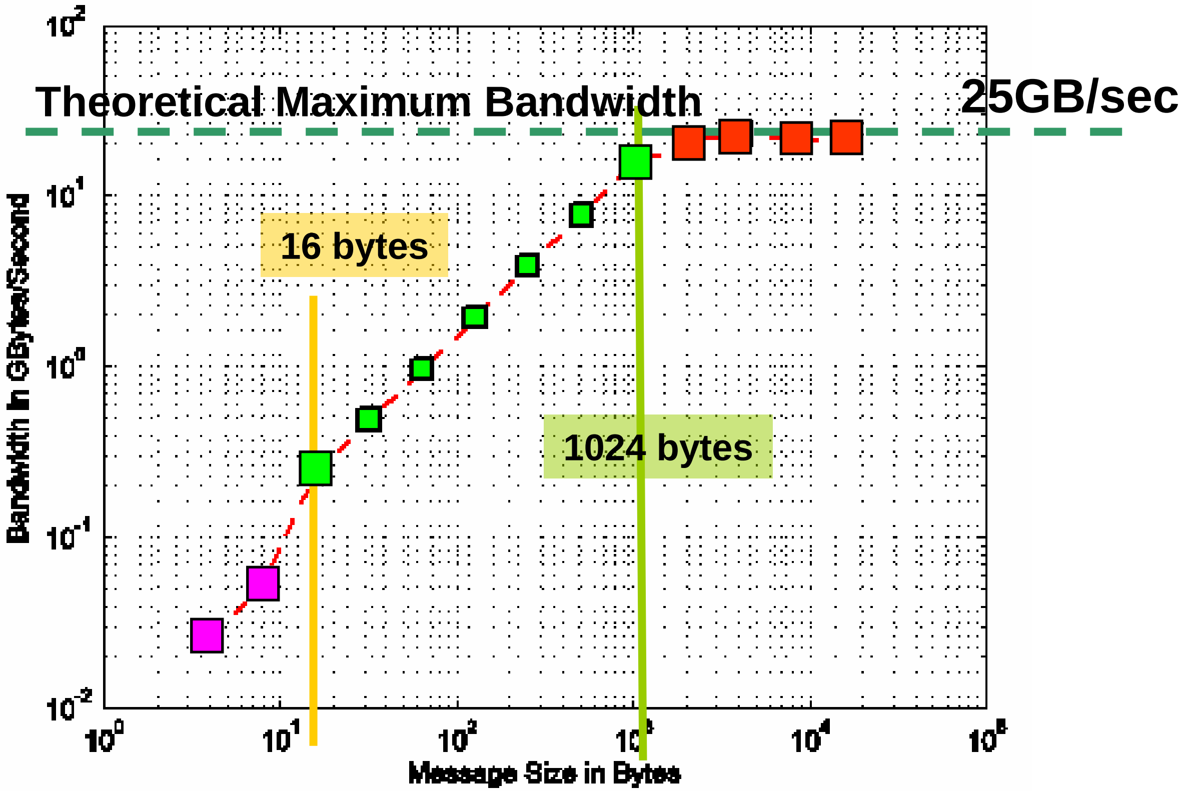
Latency for DMA put



•Significant Performance hit for memory access smaller than 16bytes

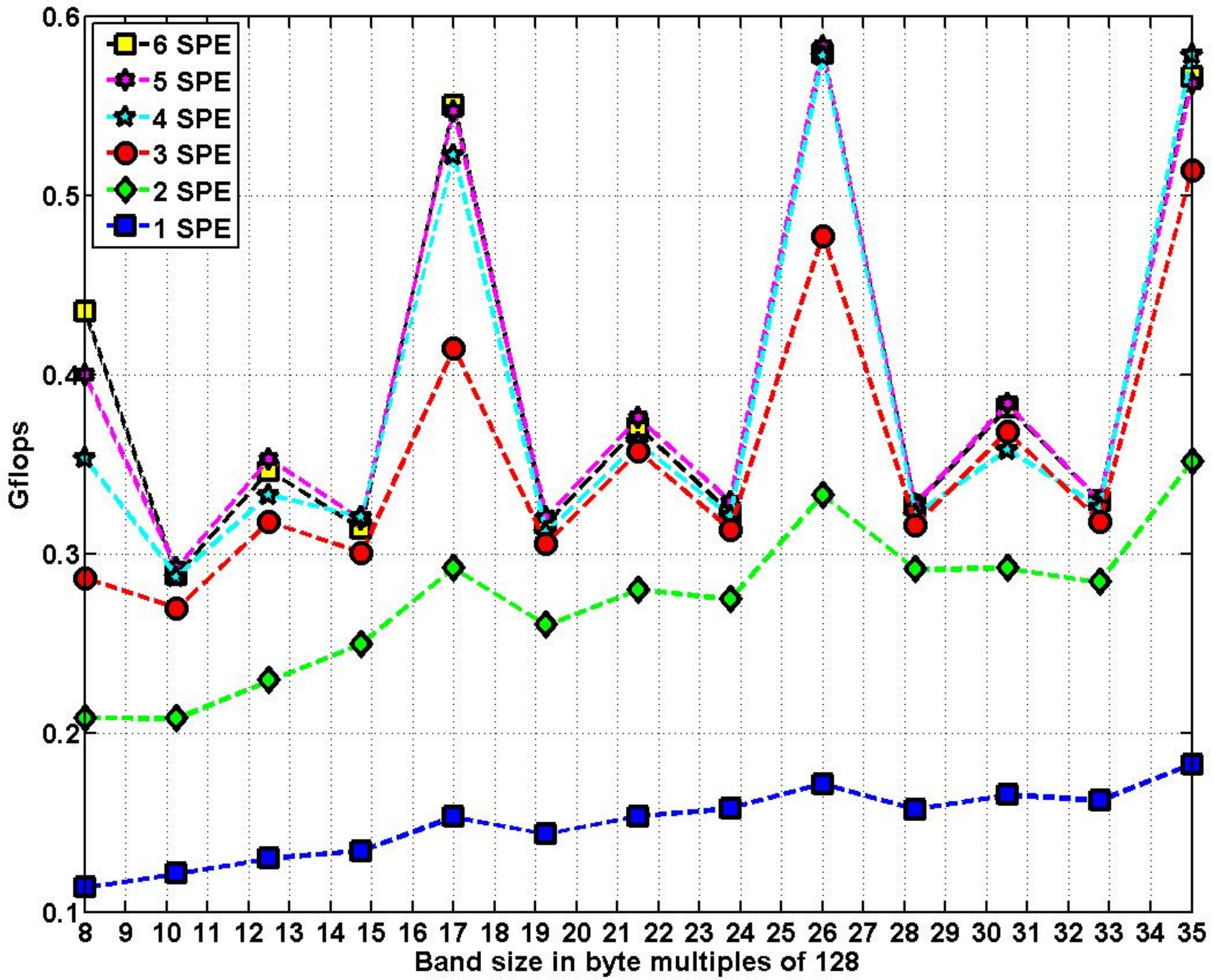
•Bandwidth limited region starts at 8x128bytes

SPE to main memory bandwidth



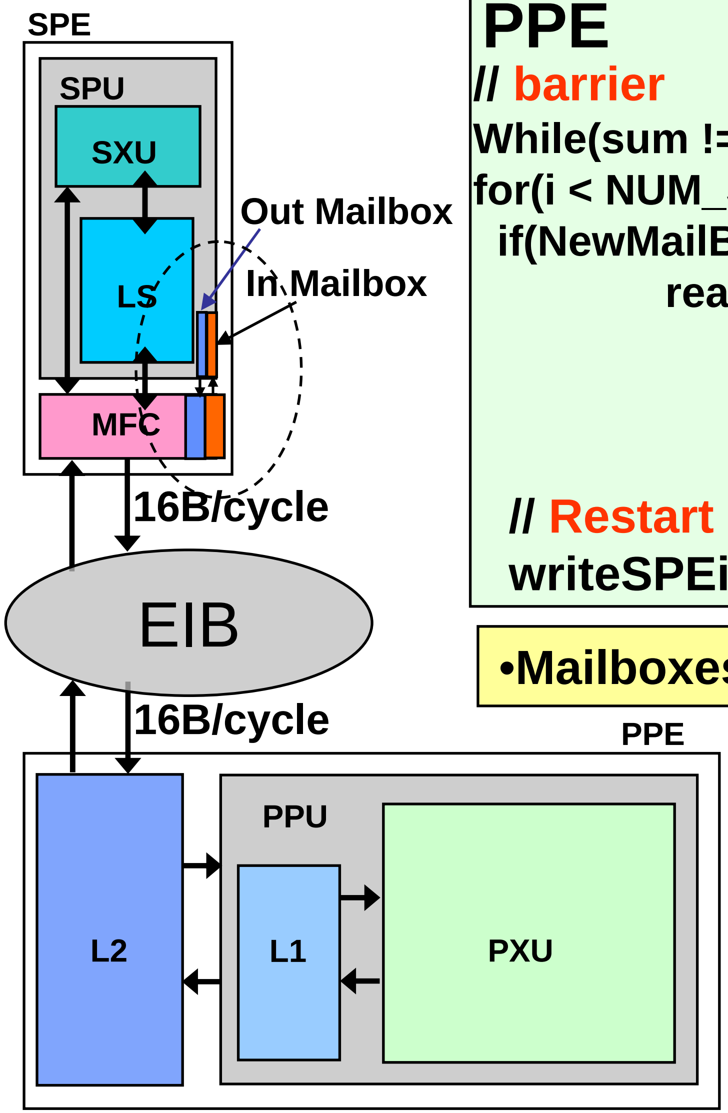
•Theoretical maximum bandwidth can almost be achieved for larger message sizes

OCA Memory Access Size Dependence



- **Out of Core performance is better when memory access is a byte multiple of 128**

PPE/SPE Synchronization Mailboxes



PPE

```
// barrier
While(sum != NUM_SPE){
for(i < NUM_SPE){
  if(NewMailBoxMessage){
              readMailBox;}}}

  // Restart SPEs
  writeSPEinMboxes();
```

SPE

```
// Notify PPE
spu_writech(SPU_WrOutMbox,1);
// Wait for PPE
spu_readch(SPU_RdInMbox);
```

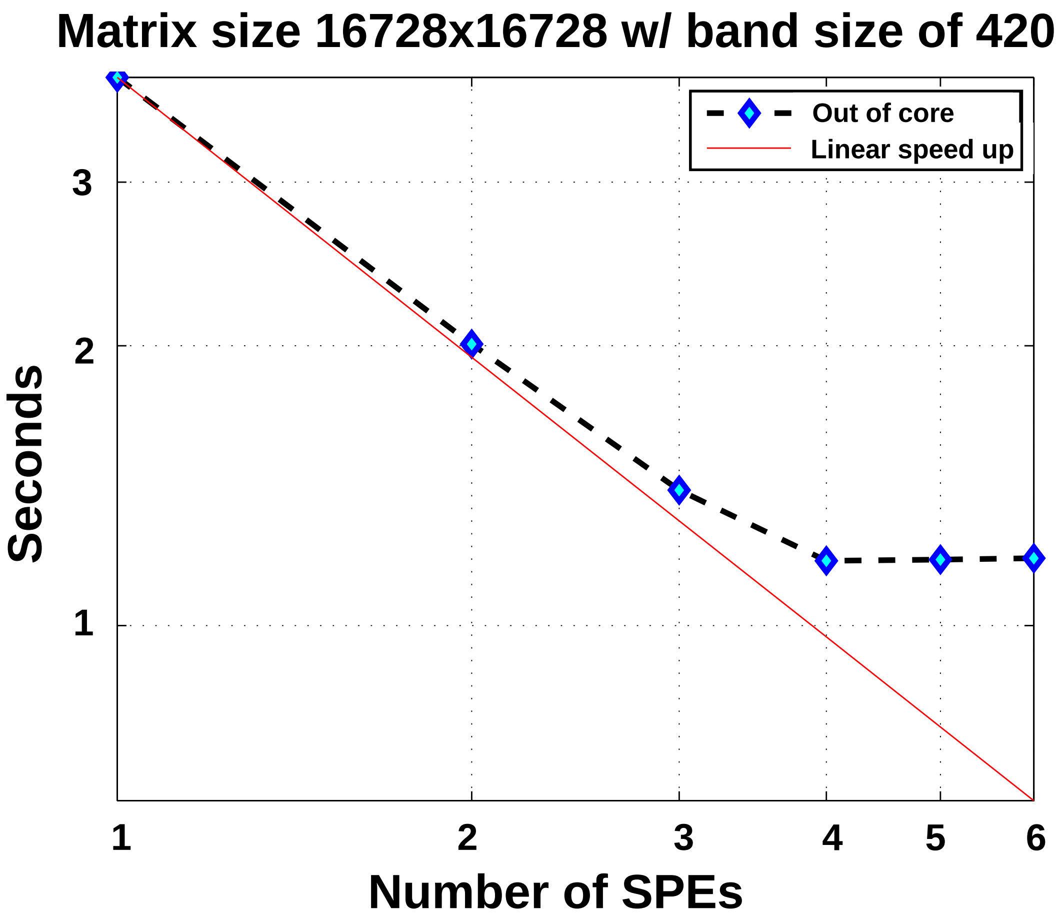
- Mailboxes are one common method of synchronization

PPE/SPE Synchronization Mailboxes Round Trip Times

• Mailboxes using IBM SDK 2.1 C-intrinsics	6.24 μseconds
• Mailboxes using IBM SDK 2.1 C-intrinsics & pointers to MMIO registers	3.65 μseconds
• Mailboxes by pointers alone	0.35 μseconds / not reliable
• Standard round trip latency 16byte message	58.33 μseconds TCP (Gigabit) 8.08 μseconds Infiniband

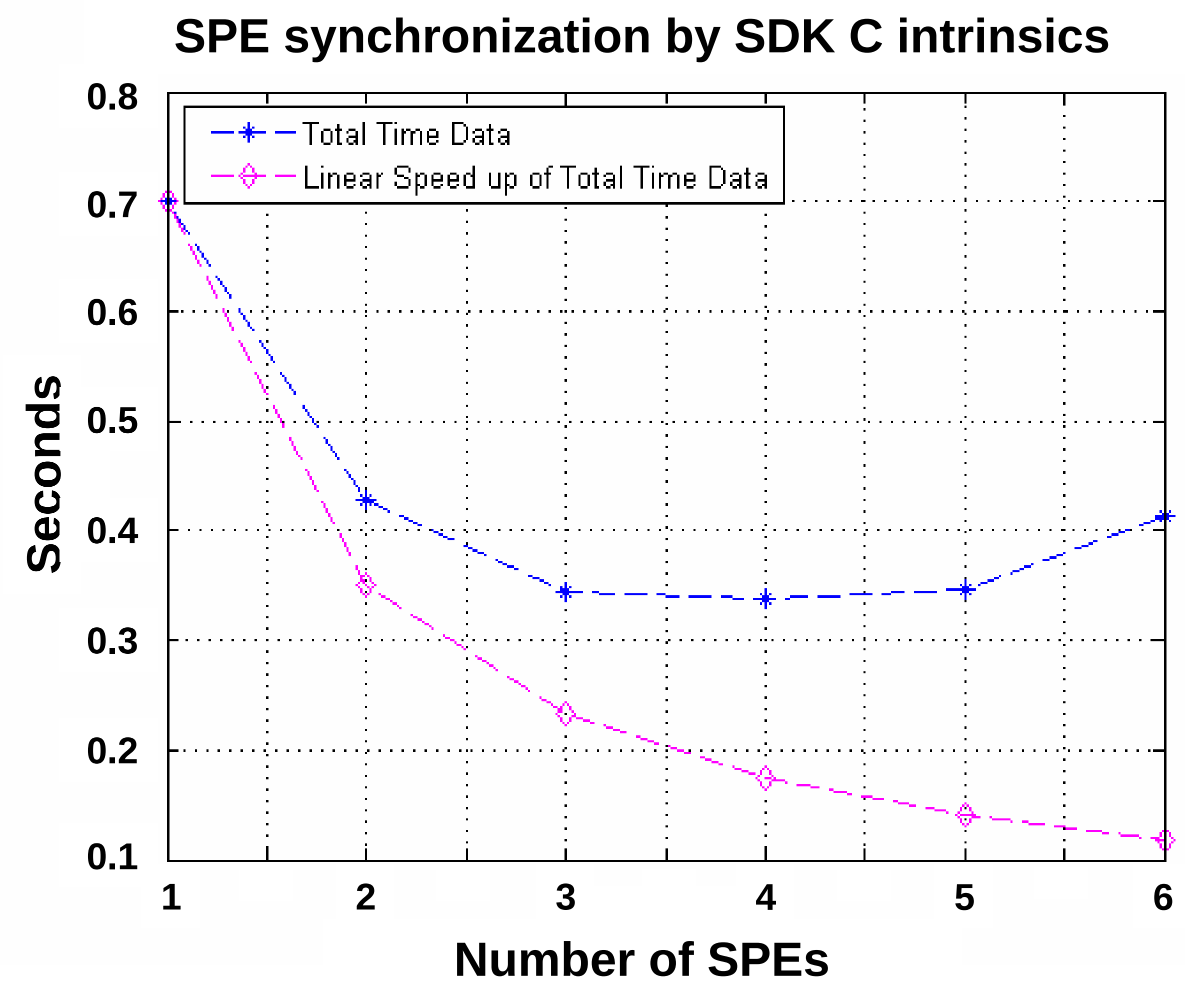
- IBM SDK 2.1 C-intrinsics for mailboxes do not seem to have idea performance.

Synchronization by hybrid of C-intrinsics and pointers to MMIO registers



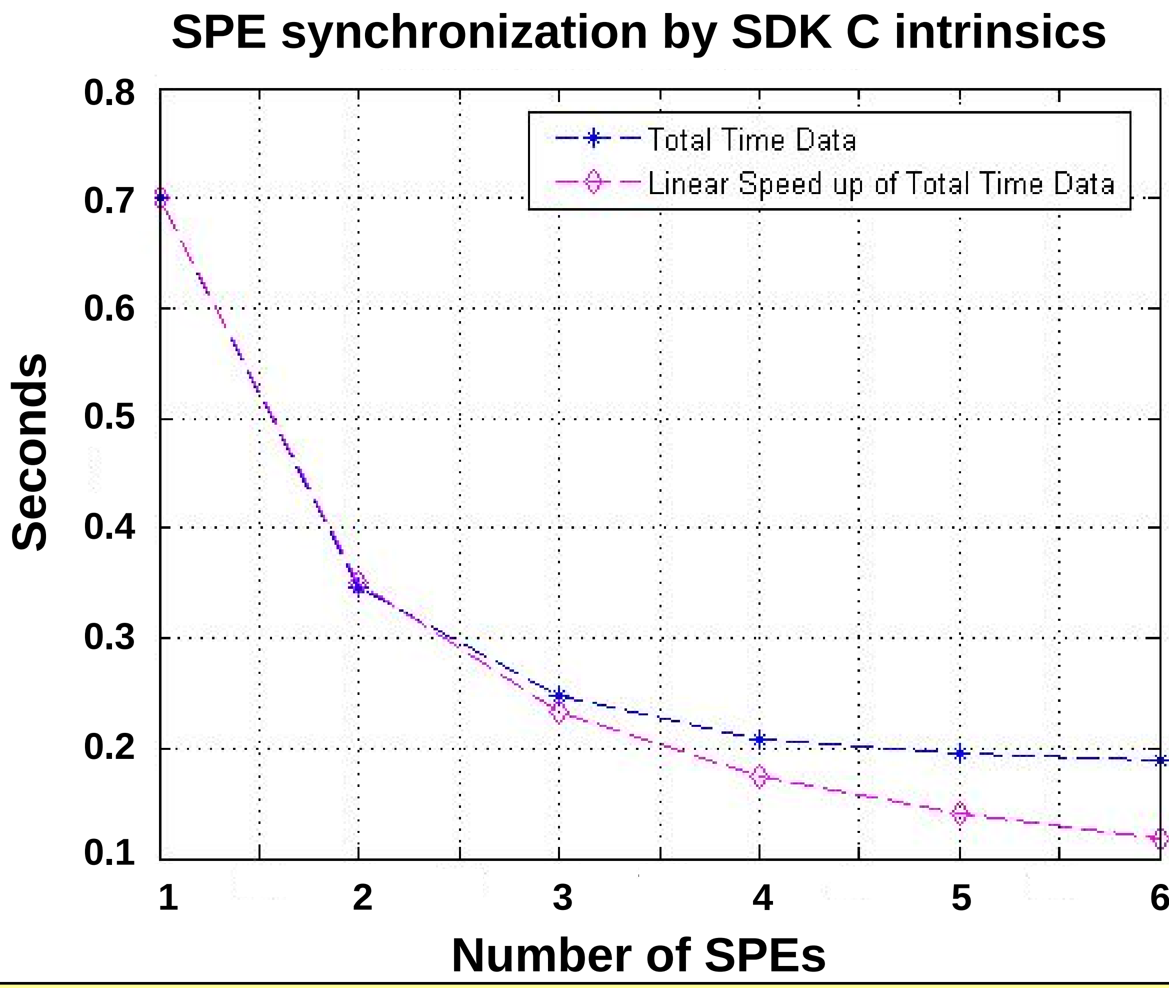
•Synchronization with IBM SDK 2.1 C-intrinsics & pointers to MMIO registers yields fairly good performance for a moderate numbers SPEs

Synchronization exclusively by IBM SDK 2.1 mailbox C intrinsics



•**Synchronization by IBM SDK 2.1 mailbox C intrinsics alone, yields little gain for low SPE count and performance LOSS after only 4 SPEs!!!**

Data from synchronization exclusively by pointers

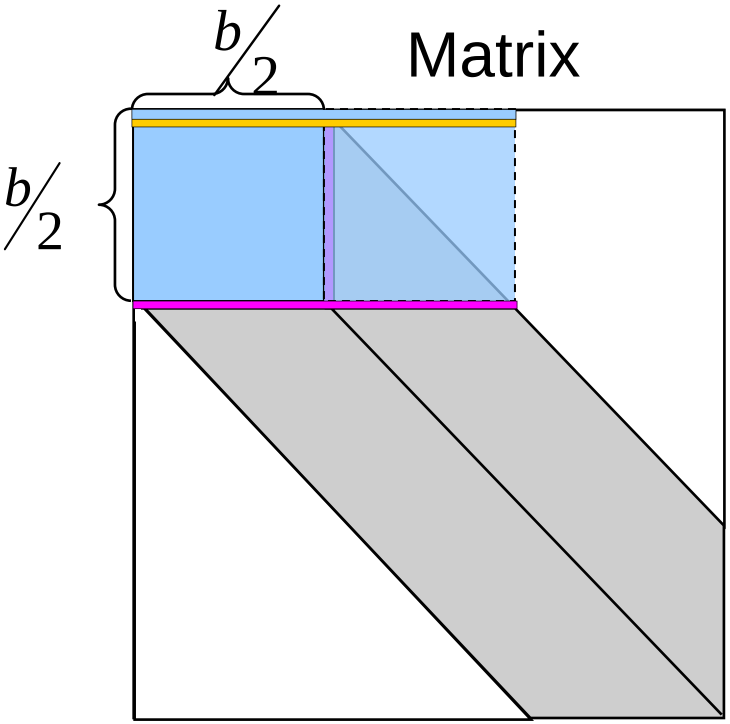


- Synchronization by pointers alone, yields a nice speed up for all SPEs
- Seems to be reliability issue with reading mailbox status register via pointers

Outline

- Introduction
- Out of Core Algorithm
- **In Core Algorithm**
  - *Hide memory accesses*
  - *Hide synchronizations*
- Summary

Banded LU
In Core Algorithm



Synchronizations N

Compute $2N\left(b/2\right)^2$

$\forall i < N$ **Multiples:** $\left(b/2\right)^2$

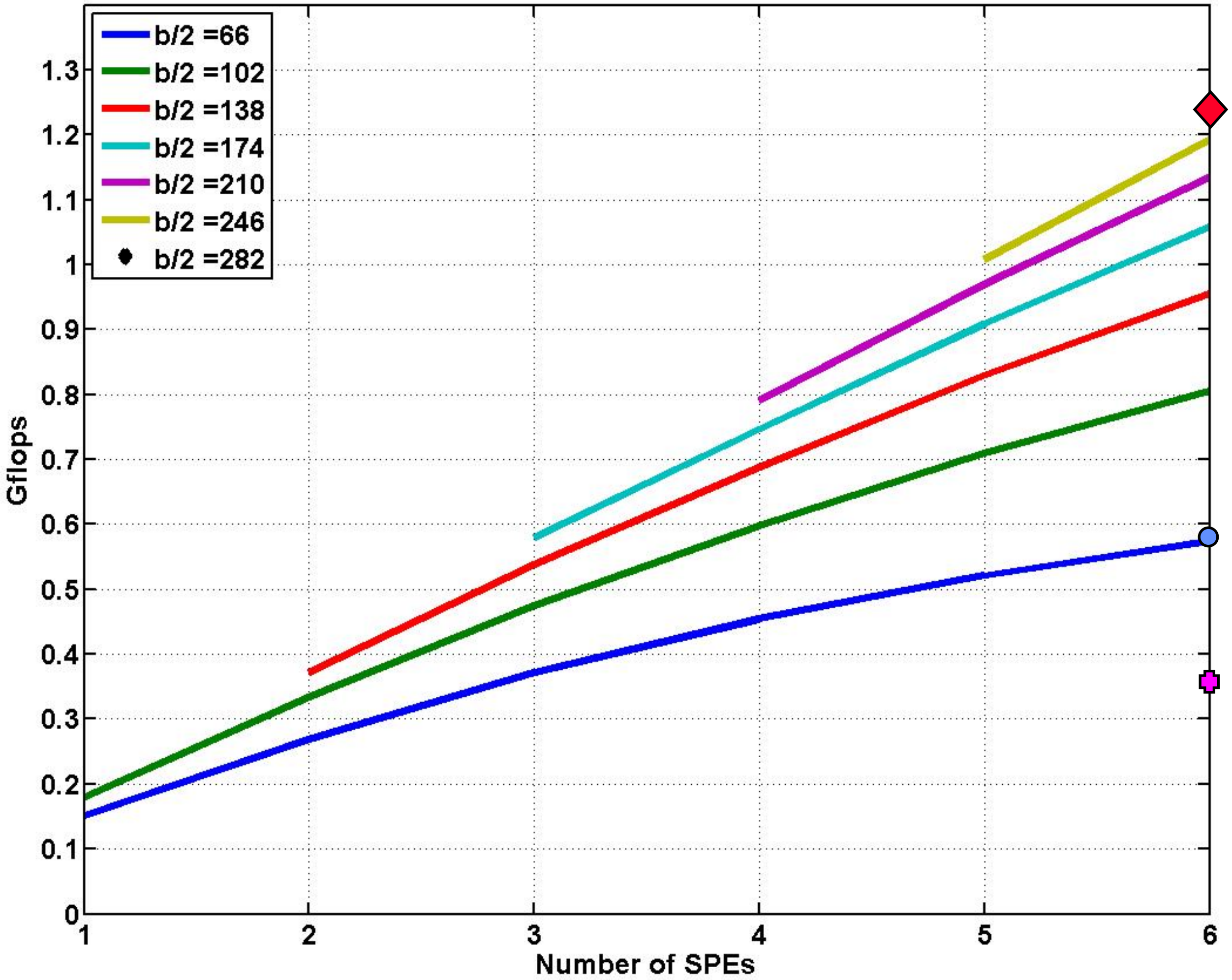
Subtracts: $\left(b/2\right)^2$

Memory Accesses		Doubles Moved
Start up Access:	1	$b\left(b/2\right)$
$\forall i < N$ **Gets:**	1	b
Puts:	2	b
Total Accesses:	$3N$	$b\left(b/2\right) + N(3*b)$

- Compute/Memory Ratio = $\dfrac{b/2}{3}$
- For a 16728x16728 matrix with band size of 420, only 0.158 GB of data would have to be moved.

In Core Performance

Achieves over 11% of theoretical performance



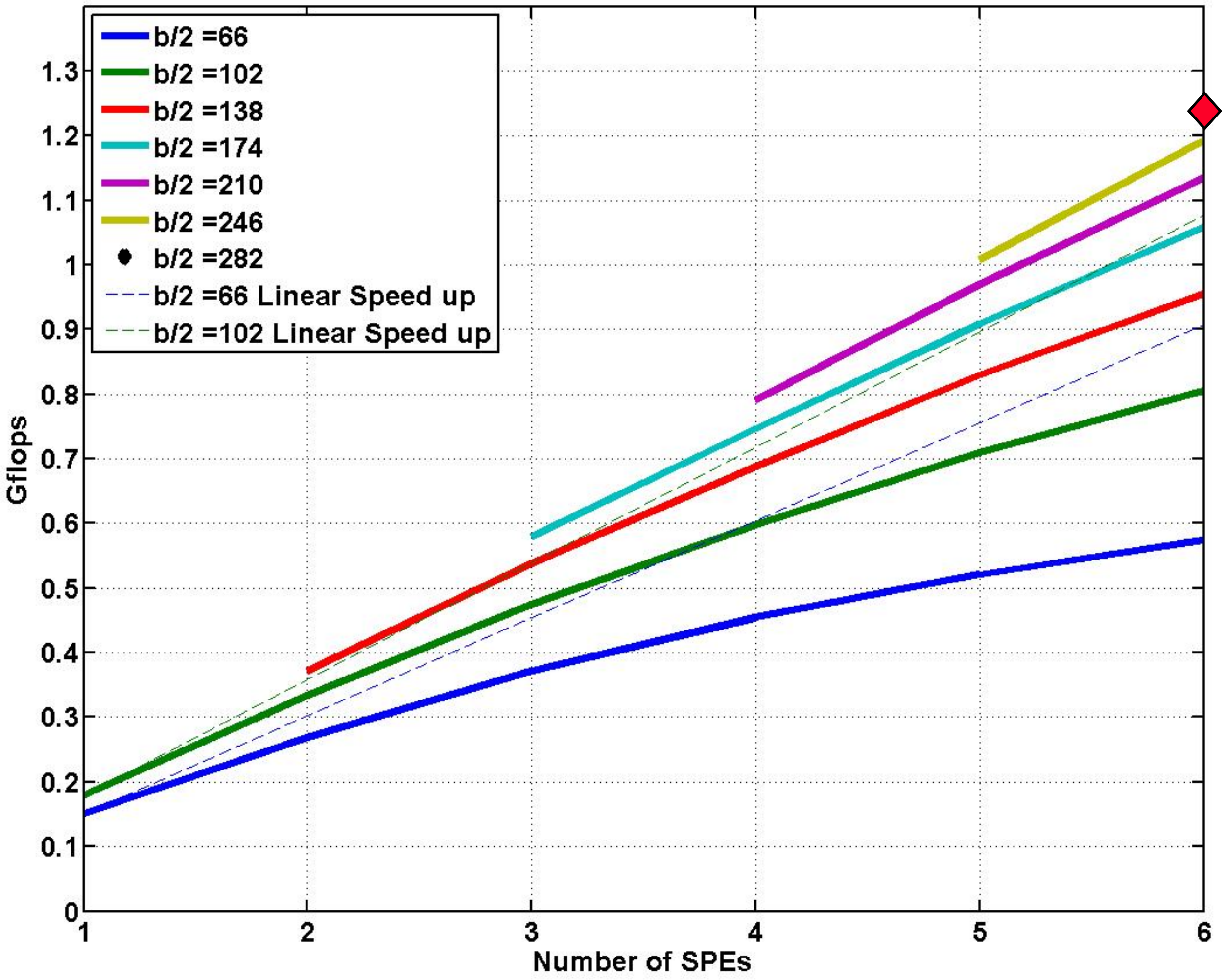
Max in core 1.23 Gflops

Max out core 0.583 Gflops

Intel Xeon 5160 3.0Ghz 0.377 Gflops

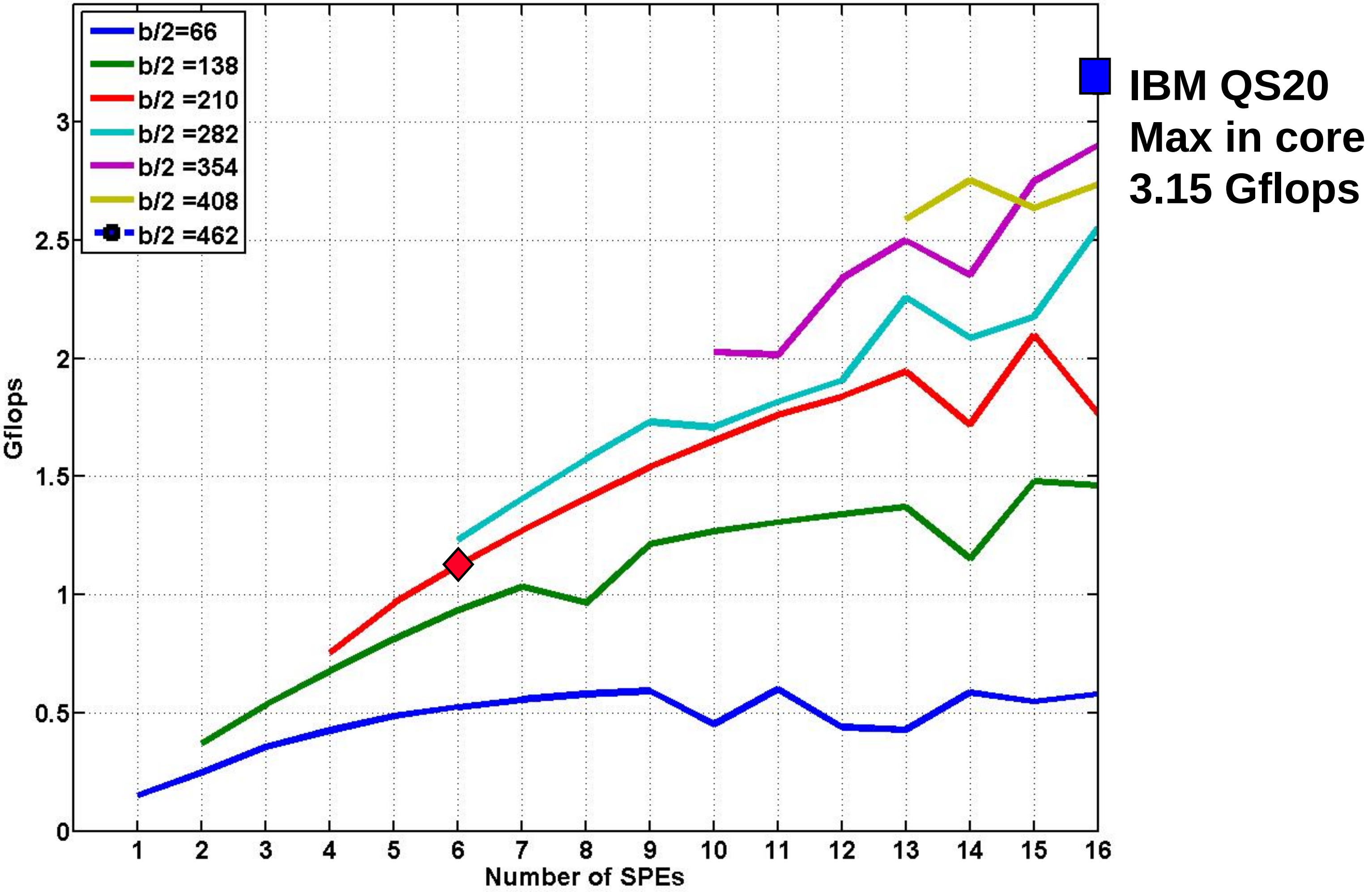
In Core Performance w/ linear speed up

Performance improves as band size increases

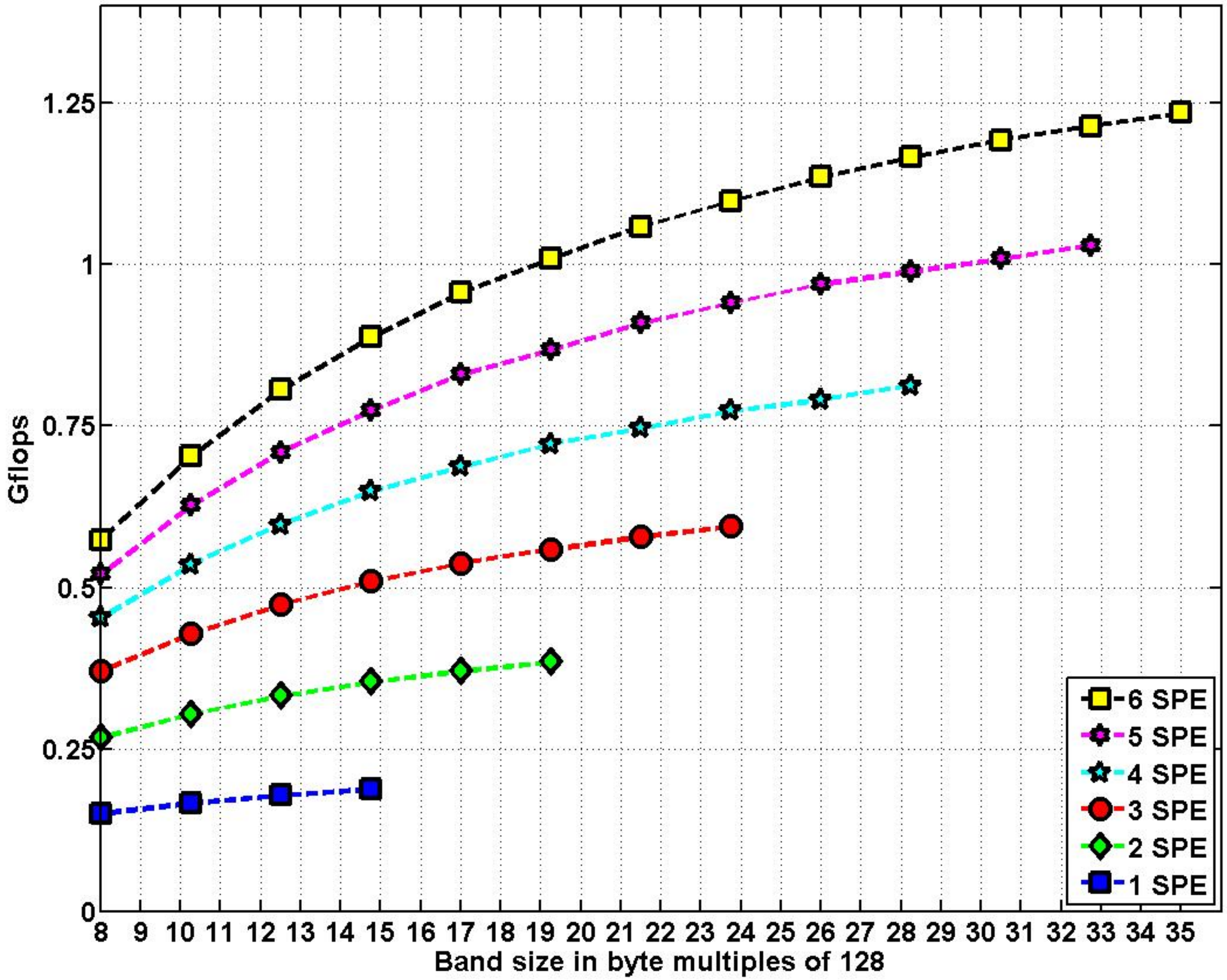


IBM QS20 performance

Achieves over 11% of theoretical performance



ICA Memory Access Size Dependence



- In core performance does not show a large dependence on memory access size

Outline

- Introduction
- Out of Core Algorithm
- In Core Algorithm
- **Summary**

Summary / Future Work

- Parallel LU decomposition can benefit from the high bandwidth of the CBE
  - Benefit depends greatly on synchronization scheme
- inCore LU offers performance advantages over out of core LU
  - Limits on size of matrix bandwidth for inCore LU.
- Partial pivoting

Acknowledgements

- **Out of Core Algorithm**
  - Sudarshan Raghunathan
  - John Chu MIT
- **In Core Algorithm**
  - Jeremy Kepner MIT LL
  - Sharon Sacco MIT LL
  - Rodric Rabbah IBM