A Characterization of Shared Data Access Patterns in UPC Programs

Christopher Barton, *Calin Cascaval*, Jose Nelson Amaral

© 2004 IBM Corporation

Outline

- **Motivation**
- **Overview of Environment**
- **Benchmarks**
- **Results**
- **Conclusions**

© 2004 IBM Corporation

PGAS Languages

- **Offer attractive programming model for large-scale machines**
- **Programmer specifies what data is shared and how it is distributed among threads**
- **Accesses to data follow shared memory-like style**
- **Compiler/runtime system manage moving shared data to ensure it is available to the accessing thread**

 © 2004 IBM Corporation

Parallel execution environments

- **Shared memory**
  - All memory locations are directly accessible, typically NUMA
- **Distributed memory**
  - Local memory locations are directly accessible, but may incur extra overheads if bookkeeping is done in the runtime
  - Remote accesses require messages
- **Hybrid**
  - Combination of shared memory and distributed memory
  - At least 3 levels of latency: local, shared and remote

PGAS languages provide a unique programming model

© 2004 IBM Corporation

Shared data access patterns

- **Understanding how shared data is accessed in a program is crucial to performance**
  - Local accesses can be privatized to improve performance
  - Blocking factor can be used to increase local accesses
- **Programs that “exchange” data with only a few threads could benefit from a hybrid architecture**
  - A group of threads maps to a truly shared address space
  - Shared data access is now a direct access for threads in the “neighborhood”, with much better latency than sending a message

© 2004 IBM Corporation

IBM xlUPC Compiler and Runtime System

- **Development version of the IBM UPC compiler and runtime system**
- **All shared variable accesses are transformed into calls to the runtime system**
- **No aggressive optimizations were enabled in the compiler**
- **The Shared Variable Directory (SVD) is used to manage allocation, deallocation and access to shared objects**

© 2004 IBM Corporation

Performance and Environment Monitoring

- **Framework**
  1. XML specification for events
  2. Tool set to generate stubs
  3. API that allows event selection and collection
  4. Runtime that implements the API
- **Manually instrumented the runtime calls to track the allocation and accesses of shared objects**

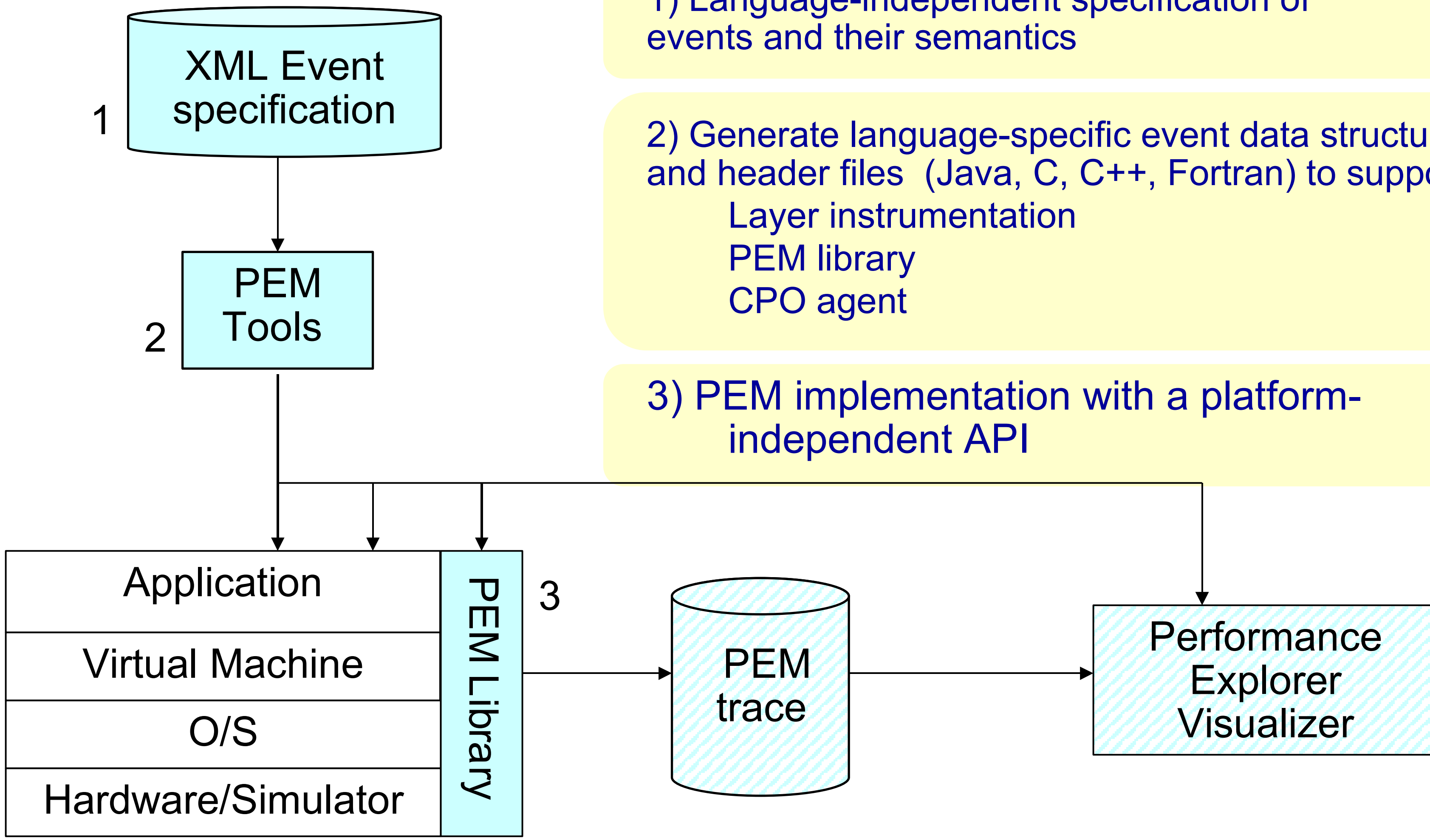
© 2004 IBM Corporation

PEM Infrastructure

1) Language-independent specification of events and their semantics

2) Generate language-specific event data structures and header files (Java, C, C++, Fortran) to support
- Layer instrumentation
- PEM library
- CPO agent

3) PEM implementation with a platform-independent API



Instrumented the xlUPC runtime to collect allocation and accesses to shared objects

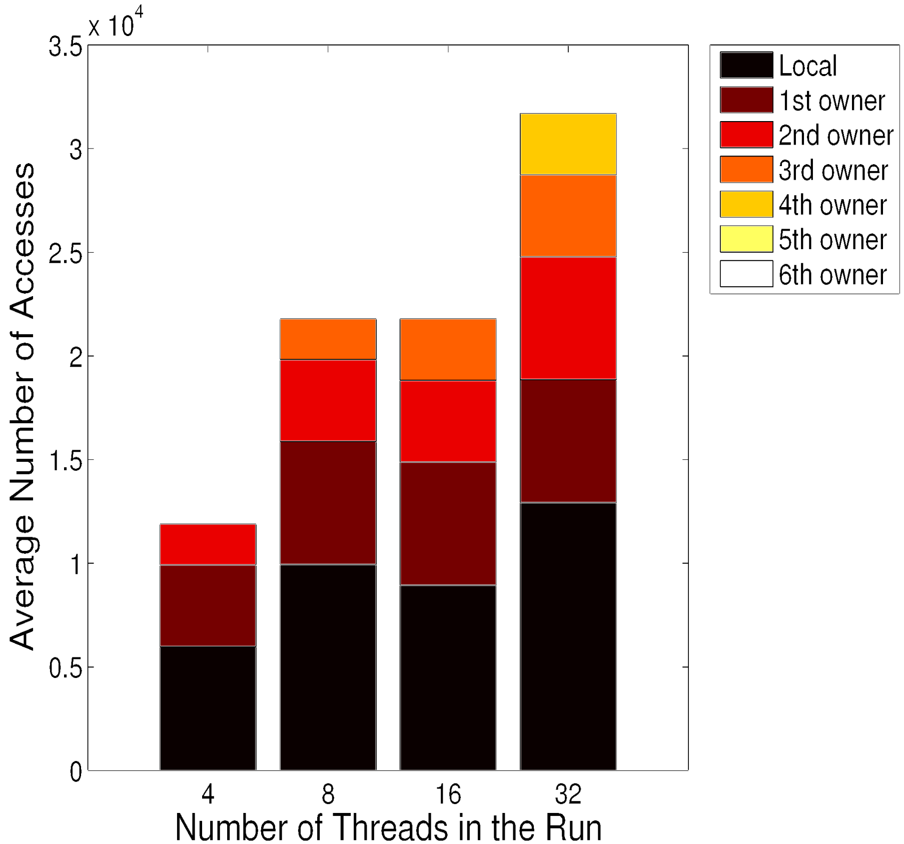
© 2004 IBM Corporation

Benchmarks

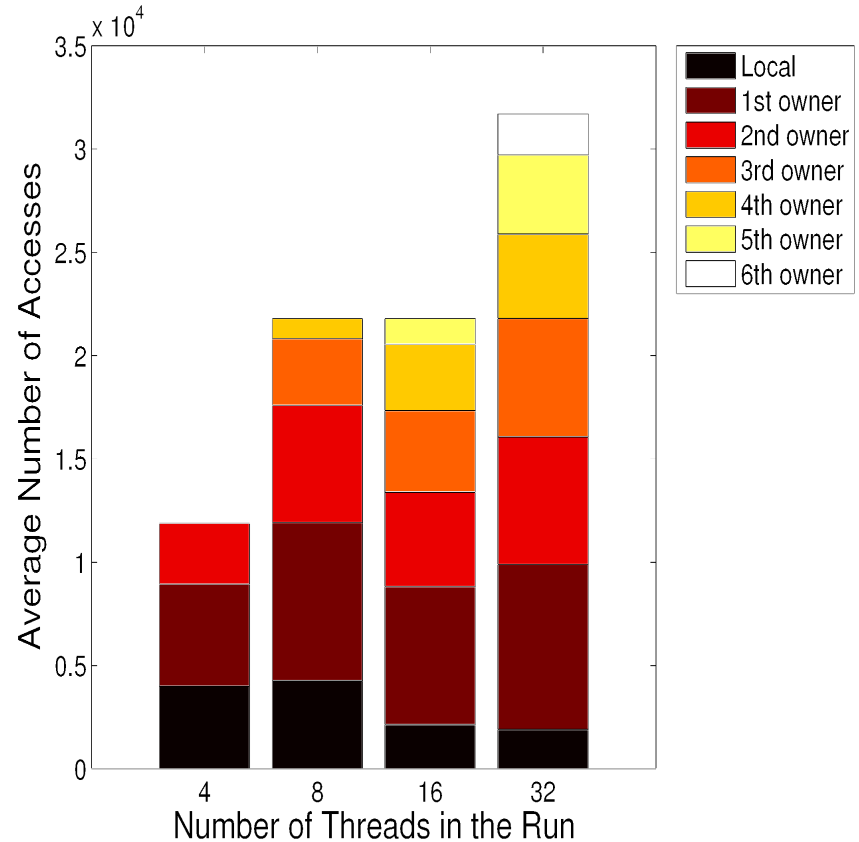
- **NAS Suite (GWU)**
  - CG
  - MG
  - IS
- **Sobel Edge Detection**

© 2004 IBM Corporation

Local-to-remote Access Ratio for CG class B



Blocking Factor = NUM_PROC_COLS



Blocking Factor = 1

© 2004 IBM Corporation

Local access ratio

Benchmark	UPC Threads	Percentage of local shared accesses				
		1TpG	2TpG	4TpG	8TpG	16TpG
CG Class B	4	50.2	83.4			
	8	45.6	72.8	90.9		
	16	41.1	68.3	86.4	90.9	
	32	40.8	59.5	78.2	90.6	93.8
IS Class S	2	50				
	4	25.1	50			
	8	13.2	25.2	50.1		
	16	7.6	13.7	25.7	50.5	
	32	6.2	9.3	15.2	27.1	51.4
MG Class S	2	74.8				
	4	62.2	74.8			
	8	55.4	62.3	74.9		
	16	52.3	56	62.3	74.9	
	32	50.6	52.9	56.1	62.5	75
Sobel Easter (BF 1)	2	26.68				
	4	23.3	60			
	8	21.7	56.7	76.7		
	16	20.8	55	73.3	85	
	32	20.4	54.1	71.7	81.7	89.2
Sobel Easter (Max BF)	2	93.2				
	4	89.7	93.2			
	8	87.7	89.7	93.2		
	16	86.2	87.7	89.7	93.2	
	32	84.3	86.2	87.7	89.7	93.2

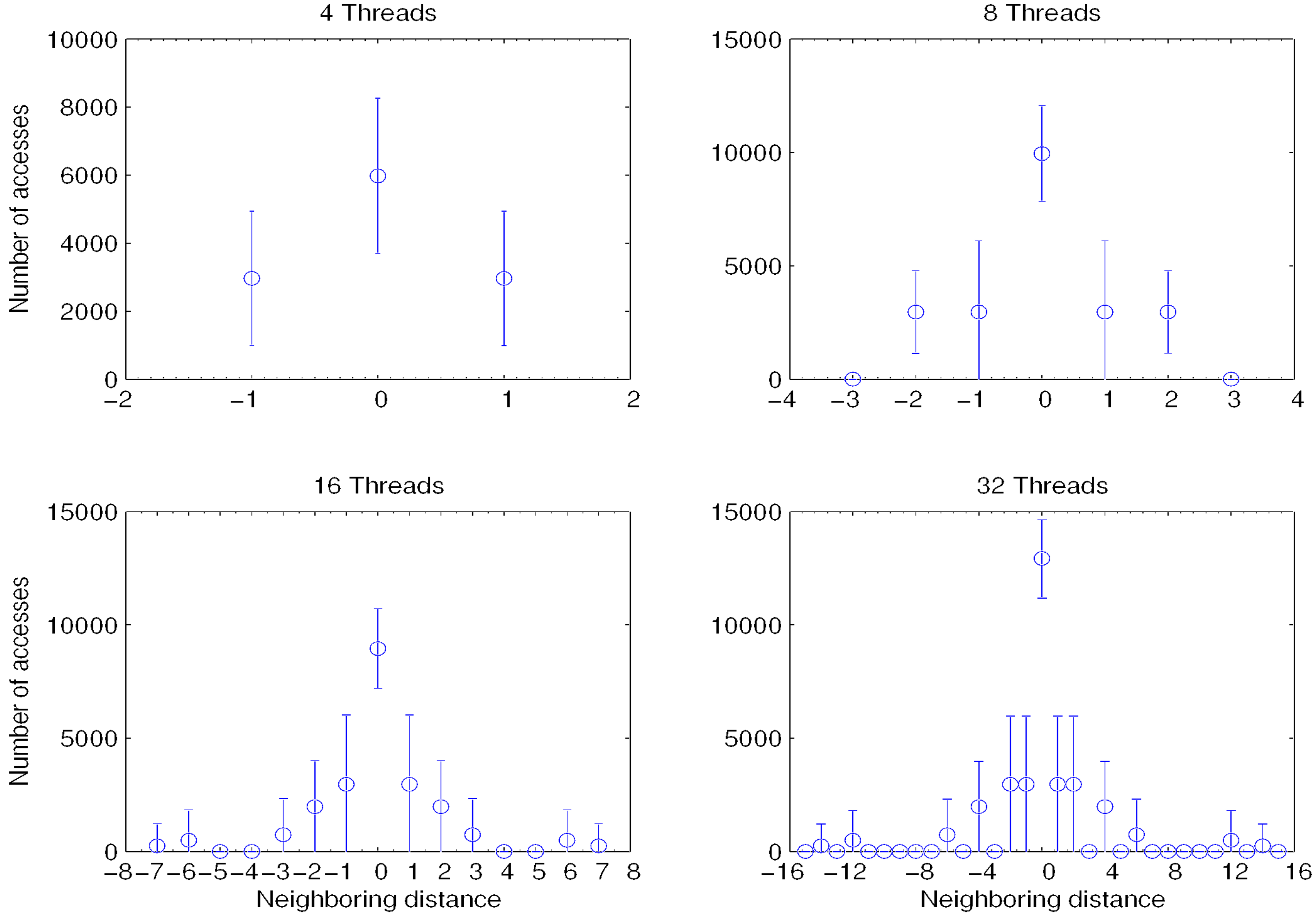
© 2004 IBM Corporation

Local-to-remote Access Ratio Lessons

- **The majority of accesses are to local data**
  - Good for performance
  - Locality optimization to reduce the translation overhead is crucial

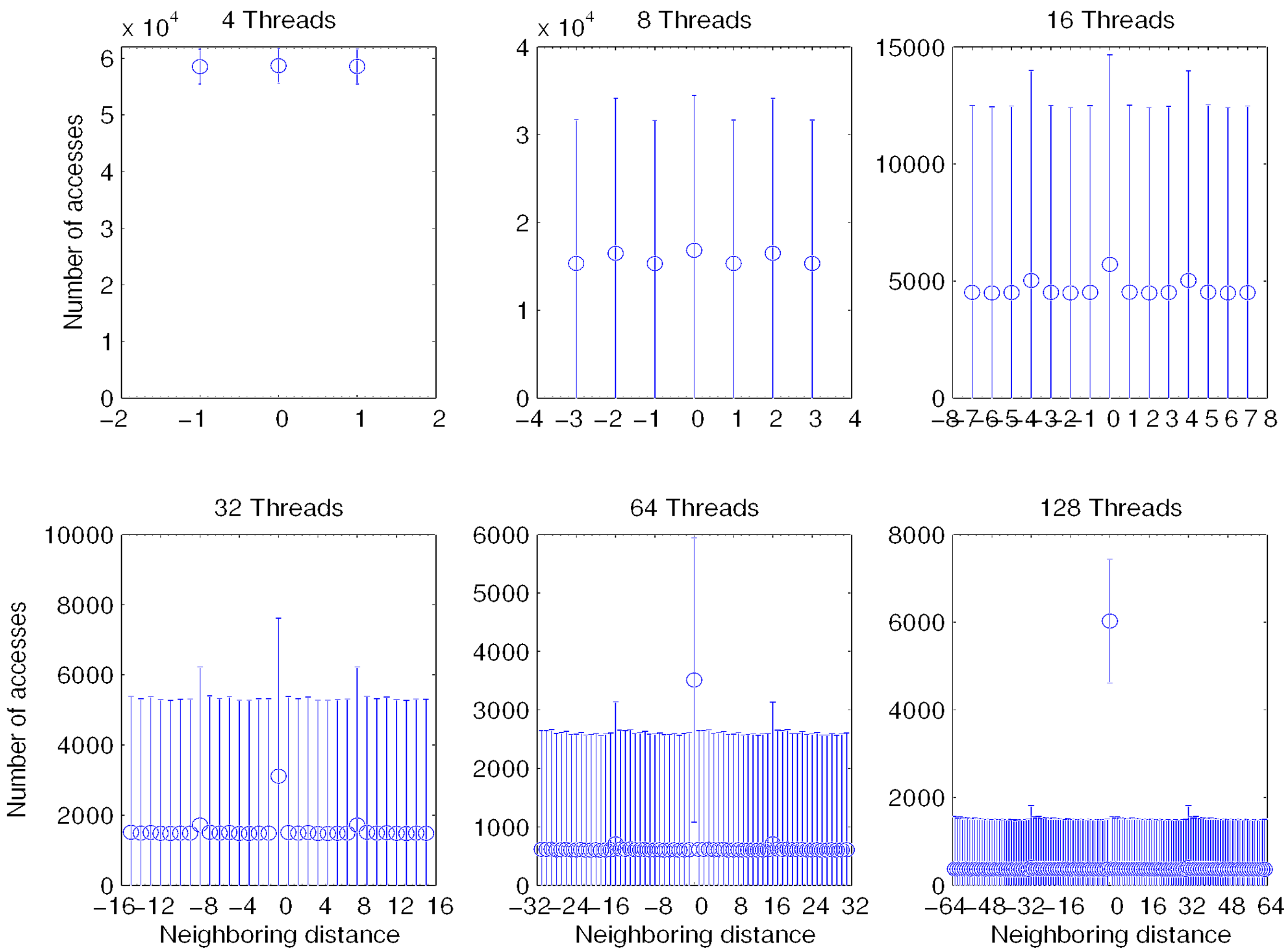
© 2004 IBM Corporation

Distance to remote accesses (CG Class B)



© 2004 IBM Corporation

Distance to remote accesses (IS Class S)



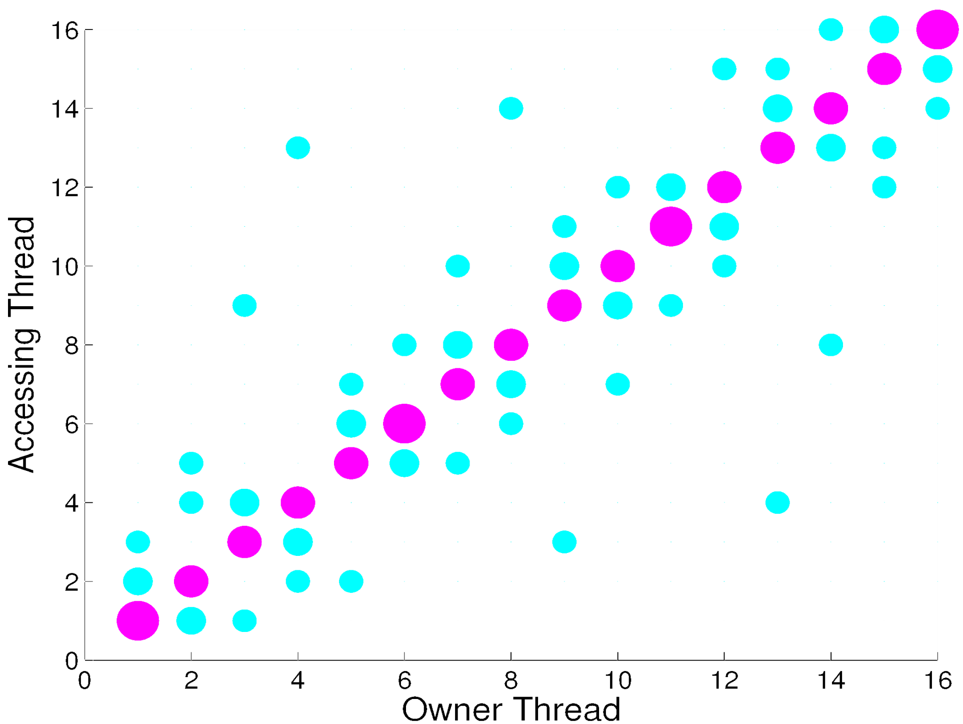
© 2004 IBM Corporation

Distance to remote data

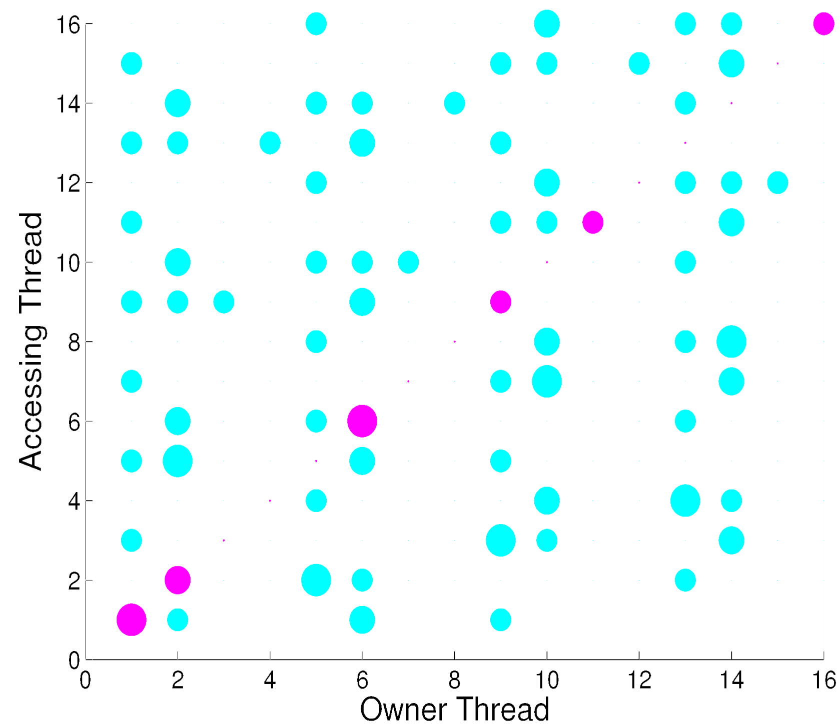
- **Each thread typically exchanges data within a small neighborhood, even when run with a relatively large number of threads (except IS)**
  – Potential to exploit hybrid architectures if mapping of threads to processors is taken into account
  – The vast majority of data can become “local”

© 2004 IBM Corporation

Distribution of shared accesses (CG Class B)



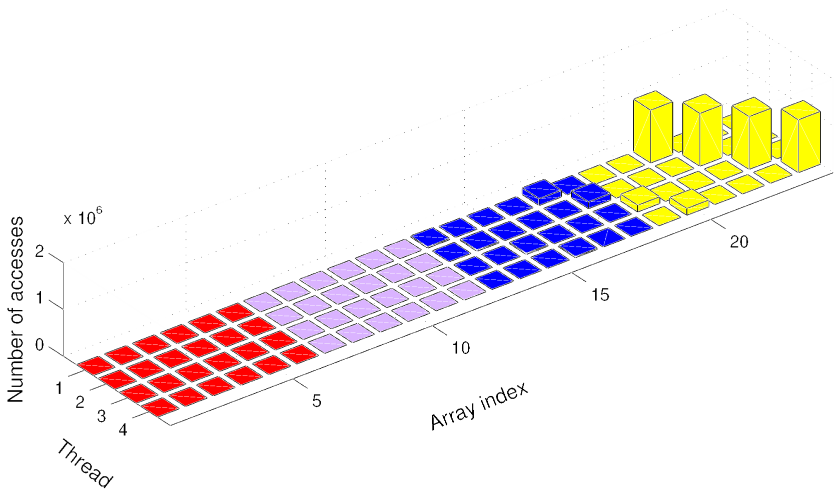
Blocking Factor = NUM_PROC_COLS



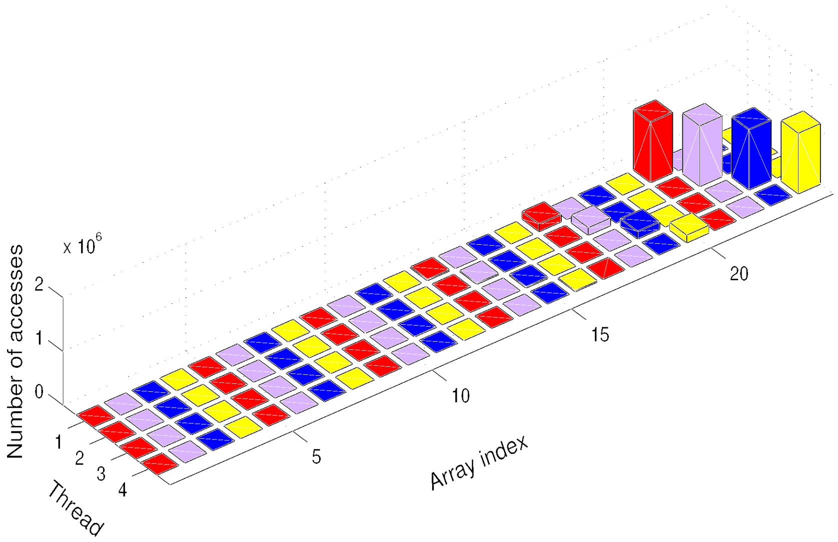
Blocking Factor = 1

© 2004 IBM Corporation

Effects of blocking factor (MG Class S)



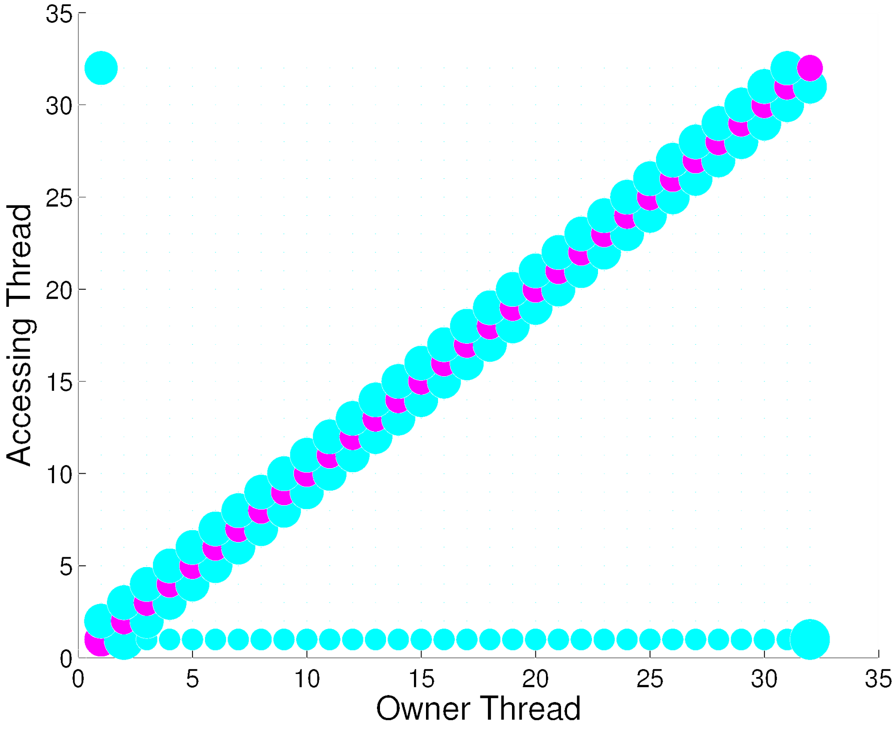
Original Blocking Factor



Blocking Factor = 1

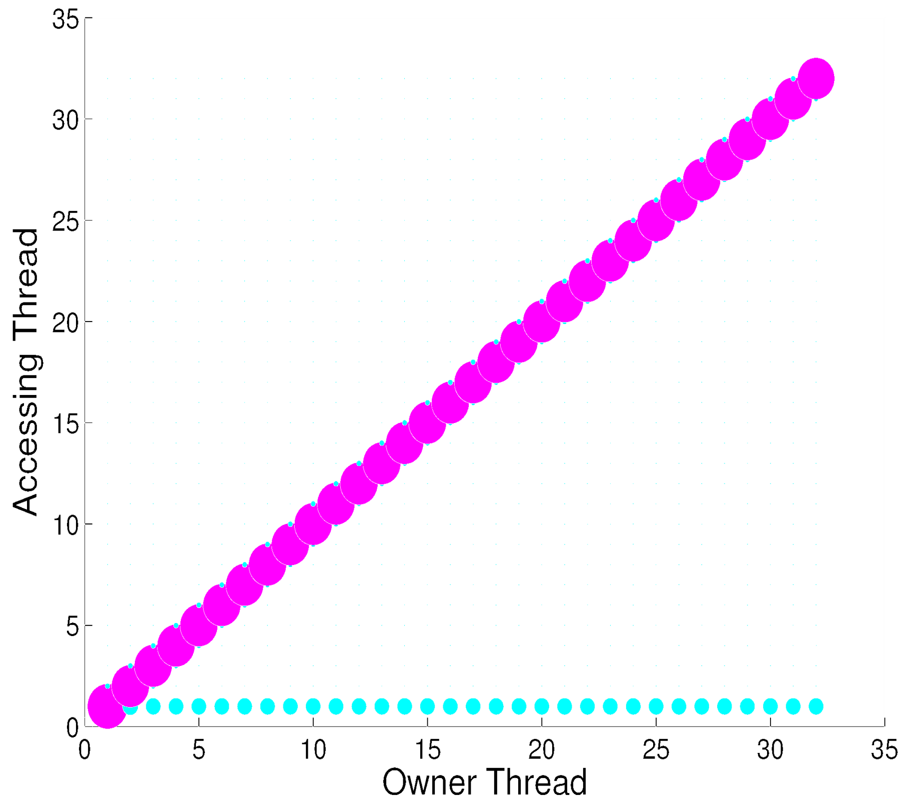
 © 2004 IBM Corporation

Effects of blocking factor (Sobel)



Blocking Factor = 1

MAX_BLOCK = (ROWS*COLUMS) / THREADS



Blocking Factor = MAX_BLOCK

© 2004 IBM Corporation

Blocking Factor

- **Semantically trivial**
- **Can have a significant effect on performance**
  - Crucial to get cache locality for single thread performance
  - Affects the amount of communication in distributed memory machines
  - Many scientific algorithms will benefit from data layout directives

© 2004 IBM Corporation

Related Work

- **Performance of UPC compared to other languages**
  - UPC vs MPI for NPB (*El-Ghazawi & Cantonnet, SC '02*)
  - Private local access vs shared local access (*Berlin et. al., LCPC '03)*
  - UPC vs CAF (*Coarfa et. al., PpoPP '05*)
  - UPC vs MPI + Pthreads (*Zhang and Seidel, IPDPS '05*)
- **Programming models for hybrid architecture**
  - Cluster OpenMP (*Hoeflinger, Intel 2006*).
  - MPI + OpenMP (*Smith & Bull, WOMPAT '00*)

© 2004 IBM Corporation

Conclusions

- **PGAS languages (UPC included) are attractive for HPC because they can provide a unique programming model for hierarchical machines**
- **Challenges:**
  - Performance on par with Fortran and MPI
  - Fix some of the peculiarities
- **Opportunities:**
  - Local accesses to shared data identifiable by the compiler
  - Small "teams" of threads that typically exchange data that will map well to hybrid architectures and increase the likelihood of local accesses
  - Layout directives that can increase locality

© 2004 IBM Corporation