Parallel Graph Processing: Prejudice and State of the Art

Assaf Eisenman[1,2], Ludmila Cherkasova[2], Guilherme Magalhaes[3], Qiong Cai[2], Paolo Faraboschi[2], Sachin Katti[1]

[1]Stanford University, [2]Hewlett Packard Labs, [3]Hewlett Packard Enterprise

Motivation

- **Large Graph Processing** is becoming increasingly important for solving multiple problems:
  - Social networks
  - Web connectivity
  - Computational Biology

- Traditional algorithms, software, and hardware **are not always effective for solving large graph problems**

- Analyze performance characteristics of graph applications
  - System bottleneck
  - Memory subsystem usage

Graph algorithms stereotypes

- Poor Scalability?
- Poor locality?
- Memory bounded: BW- or Latency-bound?

Our Profiling Approach

- **Hardware Performance Counters**
  - Core HW counters: Cache hit ratios, Stalls, etc.
  - Uncore HW counters: Memory controller memory references, LLC hit ratio, etc
- **PAPI**
  - Provides an interface for using the HW counters in the code.

Galois

- A system for automated parallelization of irregular algorithms.
- Allows the programmer to write serial C++ or Java code while still getting the performance of parallel execution
- Very efficient for large graph processing and diverse graph analytics.
- Because of its high efficiency, the main bottlenecks are system related and not code related.

Testbed, Graph Applications, Datasets

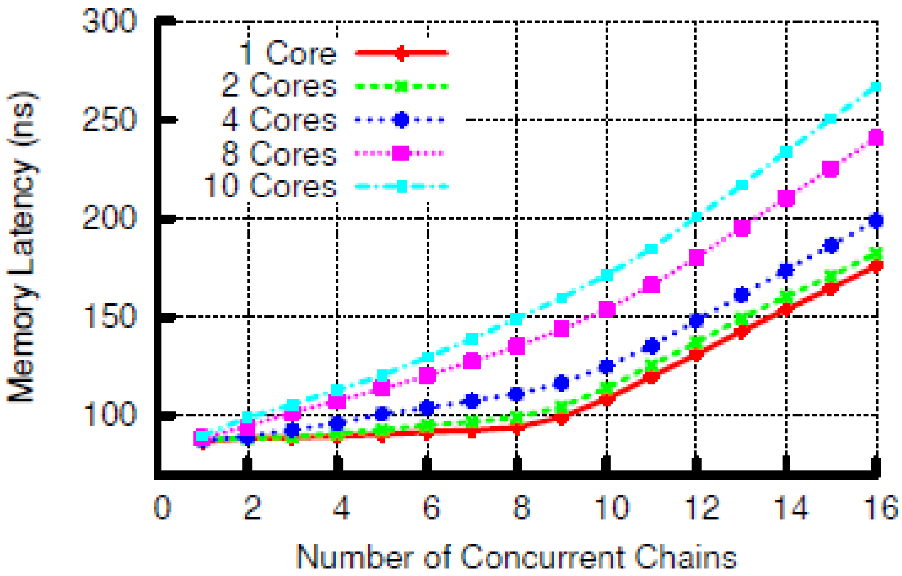
- Used Intel Xeon E5-2660 V2 with Ivy Bridge processor.
  - –10 cores per socket , frequency of 2.2 GHz, 25 MB of last level cache
- Graph Apps
  - –PageRank (**PR**)
  - –Breadth First Search (**BFS**)
  - –Betweenness Centrality (**BC**)
  - –Connected Components (**CC**)
  - –Approximate Diameter (**DIA**)
- Datasets
  - –**Twitter -** Twitter Follower Graph (61.5 M vertices, 1,458 M edges)
  - –**PLD** - Web Hyperlink Graph (39 M vertices, 623 M edges)

General system characterization

- ***pChase benchmark***
  - A well-known pointer chasing benchmark for measuring effective memory latency and bandwidth
  - Configurable number of concurrent chains of pointers to fill any desired size of memory
  - Each sequence of pointer addresses is pseudo-random, designed to defeat hardware prefetching while limiting TLB misses.
  - This access pattern *is more representative* for graph algorithms than the STREAM sequential access pattern

General system characterization

- ***Latency***
  - For 1-2 cores: growing only once core reaches 10 outstanding memory references. *Fill Buffers are a bottleneck*
  - For 4-10 cores: *Memory controller is an additional bottleneck*

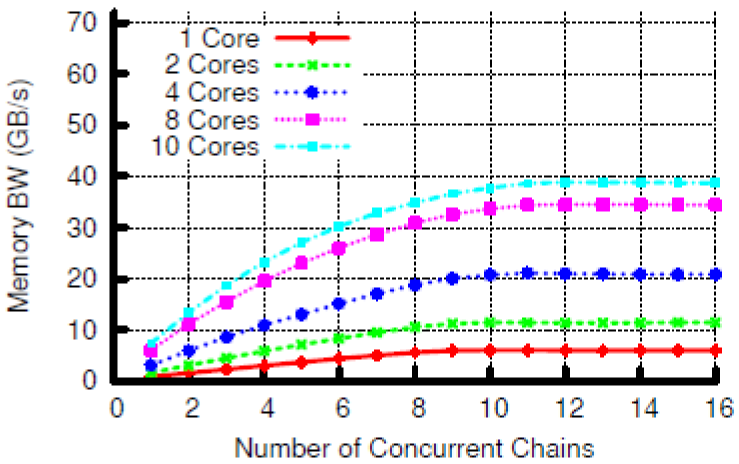


Hewlett Packard Enterprise

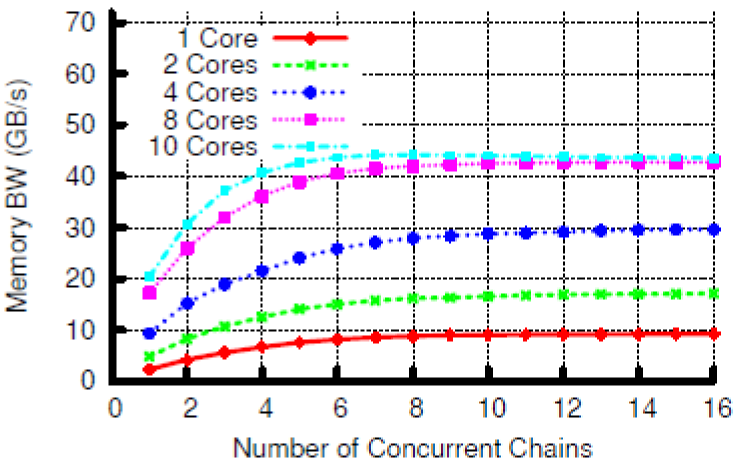
General system characterization

- ***Memory Bandwidth***
  - Memory BW scales well ***up to 4 cores*** – *Fill Buffers are a bottleneck*
  - Diminished benefits ***after that*** – *Memory controller is an additional bottleneck*

HW prefetchers ***disabled***:

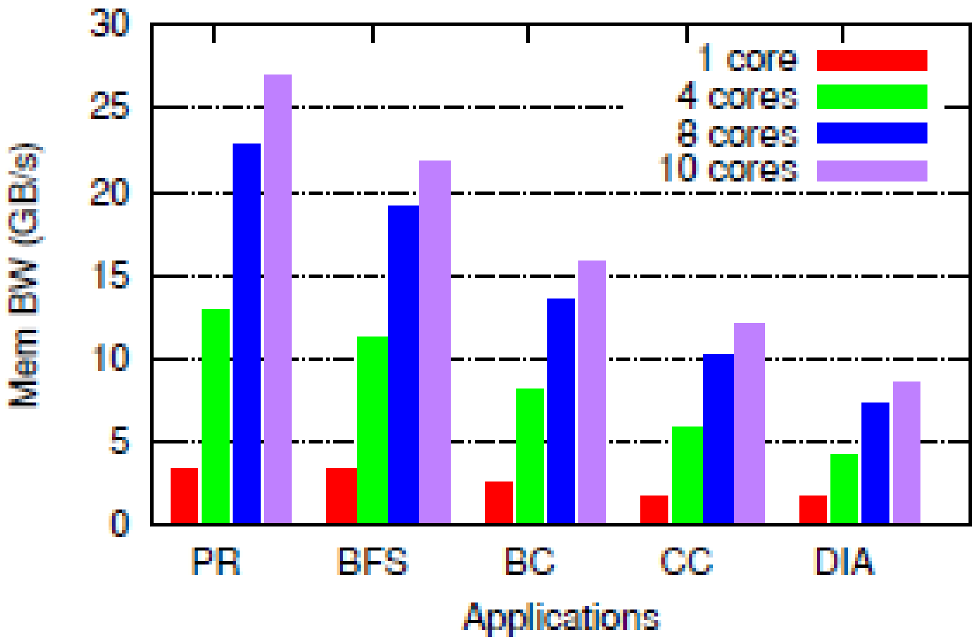


HW prefetchers ***enabled***:



Findings

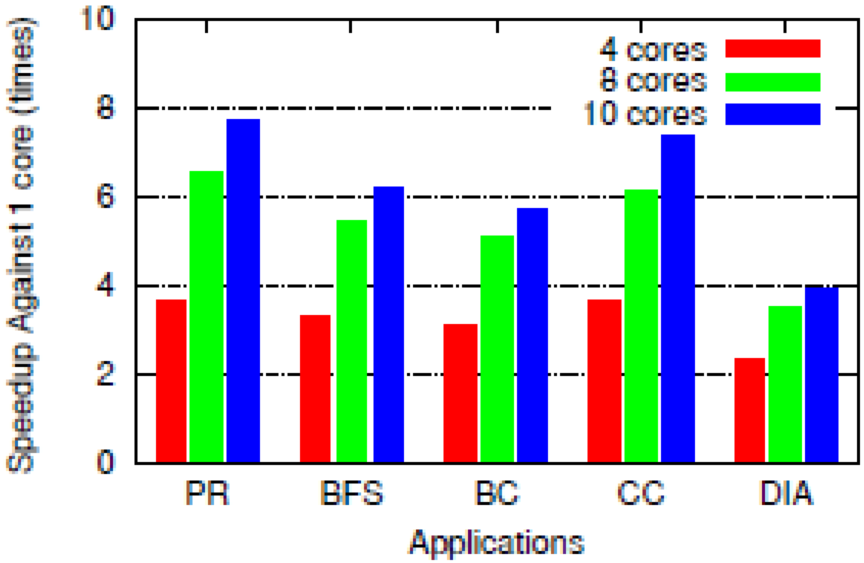
❑ ***Memory BW Scaling***

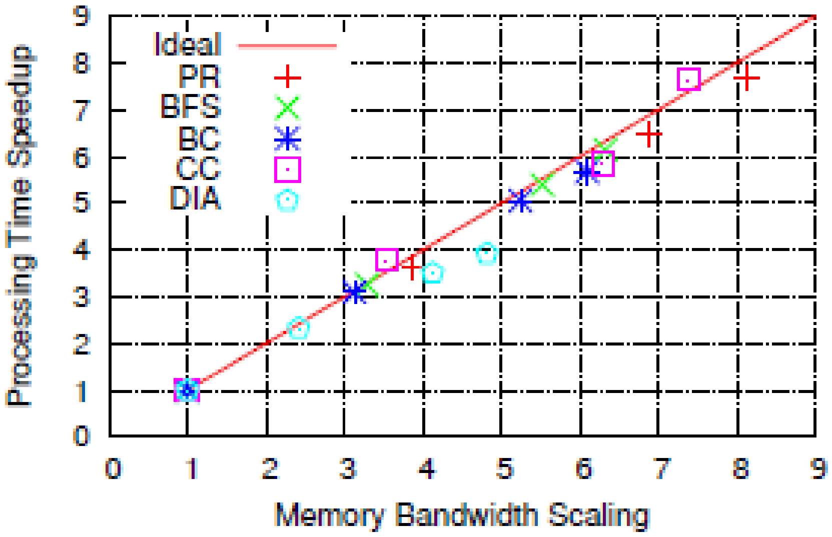


- ***Good memory BW scaling with increased number of cores***
- ***Not memory BW bounded***

Findings

❑ ***Poor Scalability?***





- ***Application speedup and scalability are highly correlated with Memory BW***

Hewlett Packard Enterprise

Findings

❑ ***Fill Buffers Occupancy and IPC***

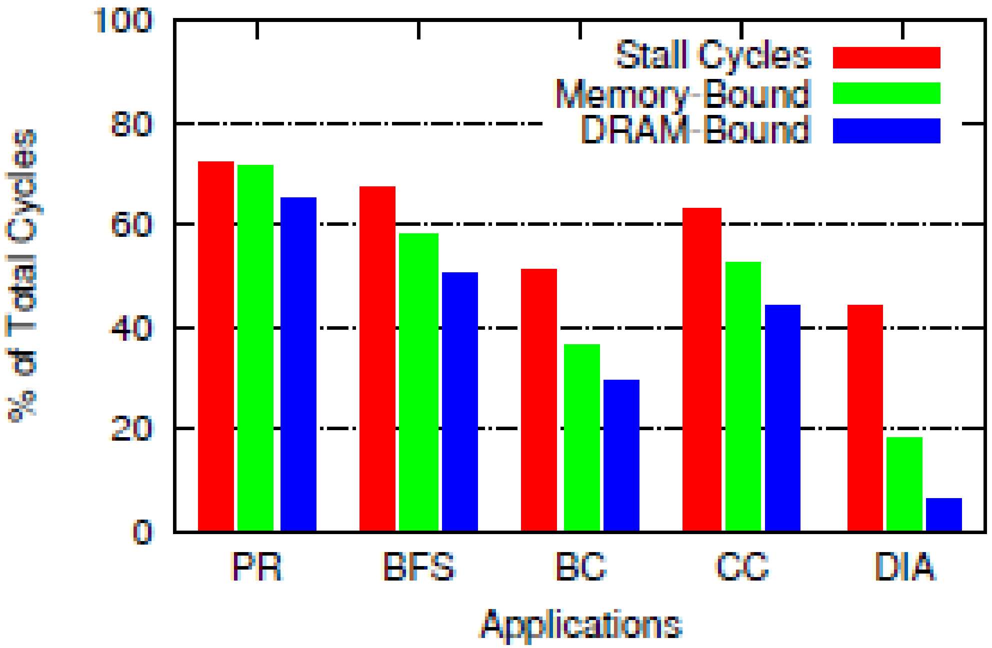
Application	Average FB occupancy
PageRank	4.7-5.5
BFS	3.3-3.5
Betweenness Centrality	1.75-2.16
Connected Components	1.37-1.55
Diameter	0.16-1

Application	IPC
PageRank	0.5-0.6
BFS	0.5-0.8
Betweenness Centrality	0.6-0.9
Connected Components	0.7-1
Diameter	0.7-1.2

- ***Fill Buffers are not a bottleneck***
- ***IPC numbers are low***

Findings

❑ ***Then what are the system bottlenecks?***



- ***Memory latency bound!***

Hewlett Packard Enterprise

Findings

❑ ***Poor locality?***

Application	L1 Hit Rates	LLC Hit Rates
PageRank	74-77%	35-39%
BFS	89-90%	34-37%
Betweenness Centrality	93-98%	30%-33%
Connected Components	95-96%	29%-31%
Diameter	96-98%	10%-22%

- ***Significant cache hit rates***

Graph Algorithms - Conclusions

- Good Scalability
- Significant locality
- Memory BW is not fully utilized
- FB are not fully utilized
- Mostly memory latency bounded

Thank you!

Questions?

Hewlett Packard Enterprise