

Exploring Memory Management Strategies in Catamount

Kurt Ferreira, Kevin Pedretti, and Ron Brightwell
Scalable System Software Group
Sandia National Laboratories

Cray Users Group
Helsinki, Finland
May 8, 2008



What to Expect

- **Description of phenomenon we've observed using the STREAM micro-benchmark**
 - Large memory bandwidth swings based on memory layout
 - Comparisons to Cray Linux Environment (CLE / CNL)
- **Due to level of locality you probably aren't aware of**
 - Hopefully interesting
 - Possibly useful
- **Mitigation techniques we're working on that alleviate issue while maintaining LWK advantages**
 - Predictable memory layout
 - Simple network stack (no pinning/unpinning)

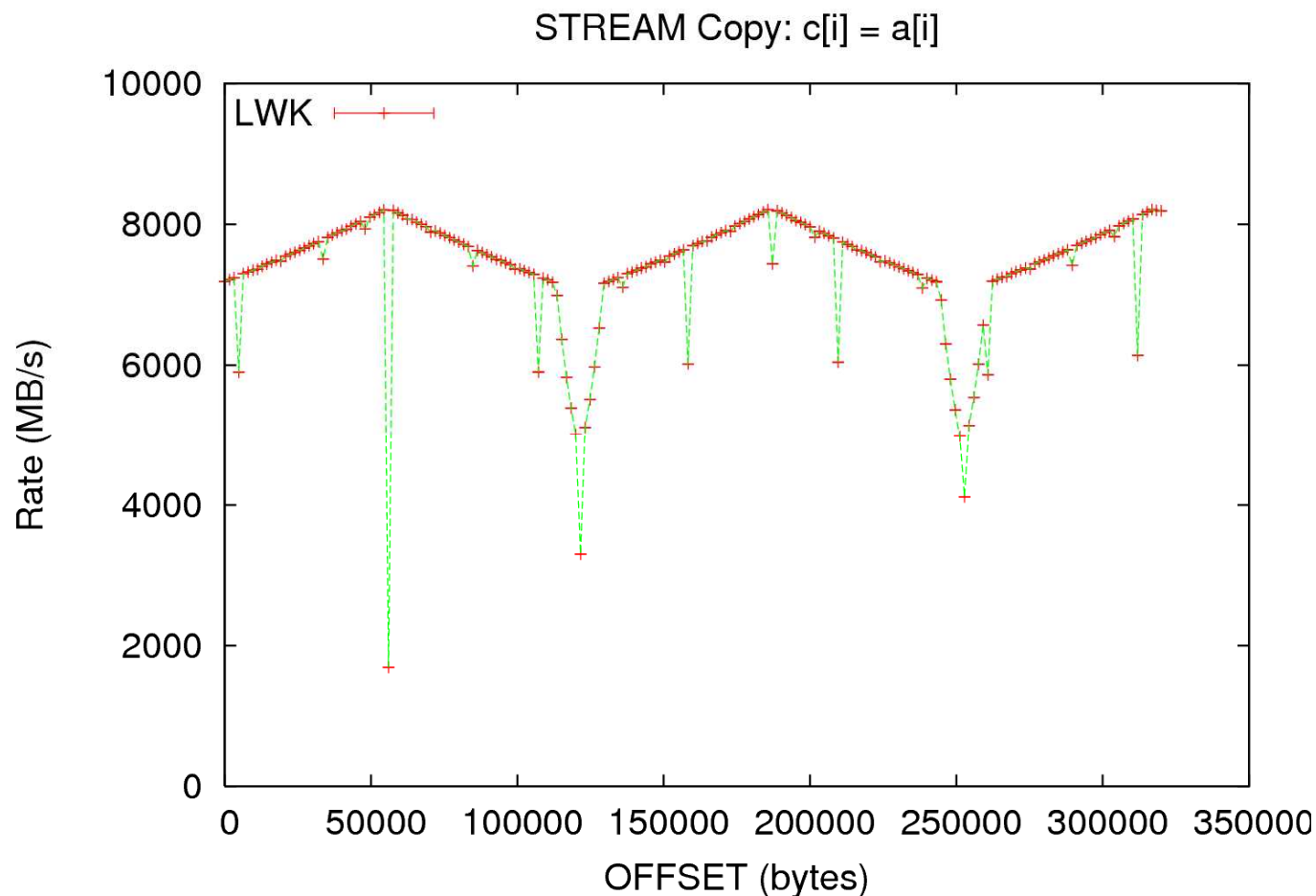


STREAM Benchmark

- Old benchmark, now component of HPCC
- Four memory intensive kernels over arrays of doubles:
 - Copy: $a[i] = b[i]$
 - Scale: $a[i] = \text{scalar} * b[i]$
 - Add: $a[i] = b[i] + c[i]$
 - Triad: $a[i] = b[i] + \text{scalar} * c[i]$
- OFFSET define controls spacing/alignment of arrays in memory:

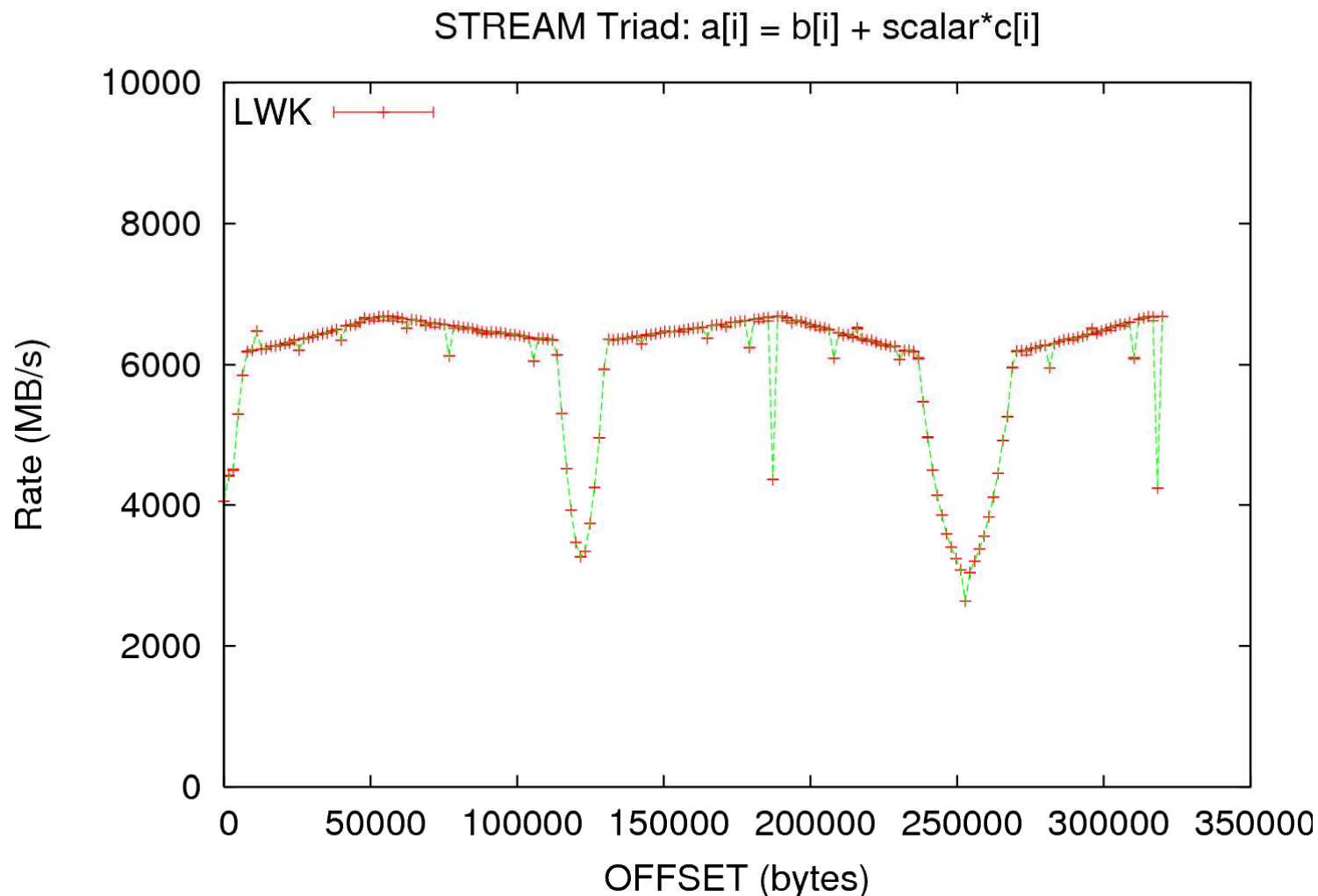


Mysterious STREAM Copy Sawtooth on Catamount



$N=2000000$, ~16MB arrays

STREAM Scale, Add, and Triad Similar

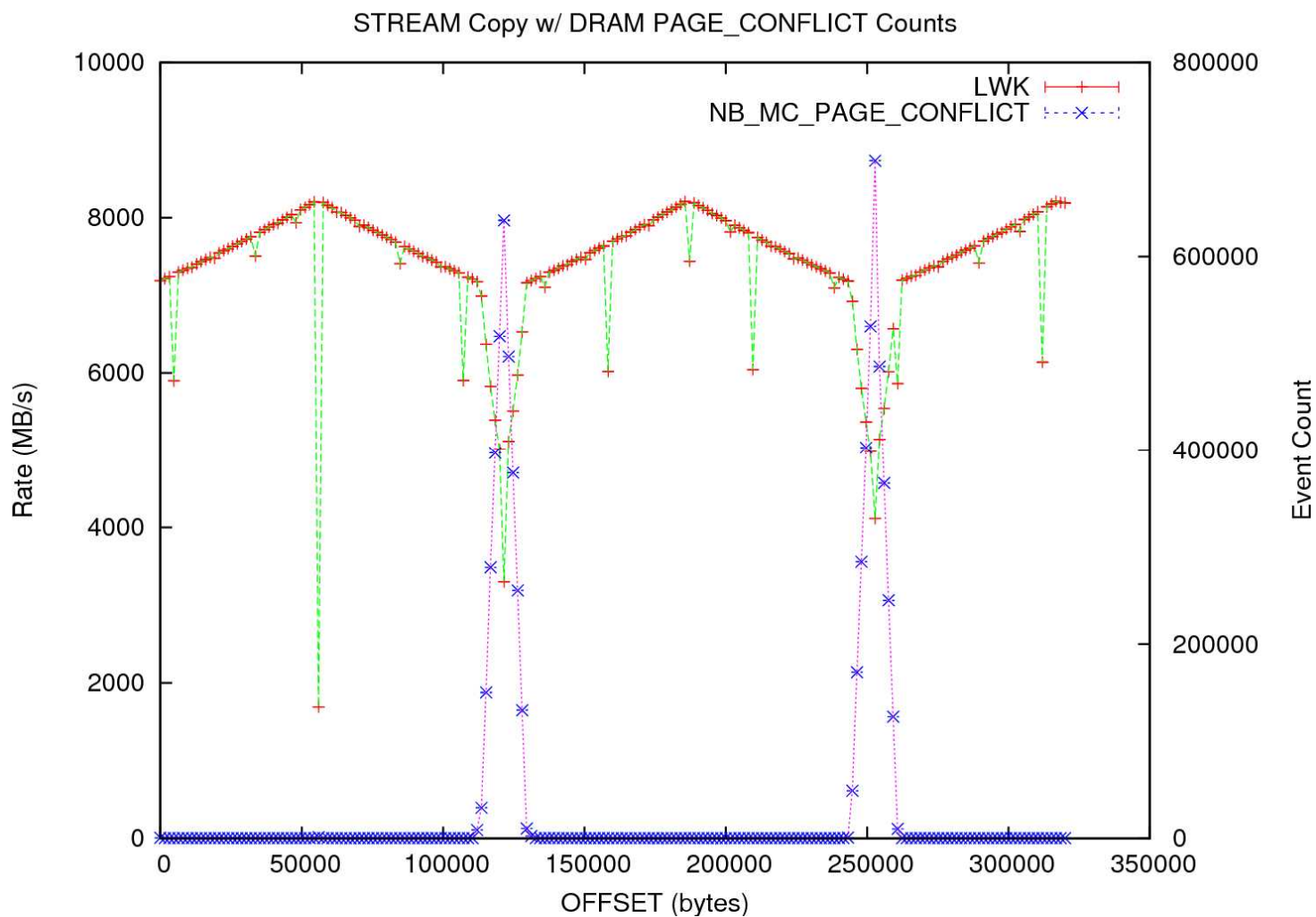




What's Going On?

- **Mystery for 2+ years**
 - First observed by Courtenay Vaughan while gathering Red Storm HPCC results
 - Careful tuning performed to avoid valleys
- **Suspects:**
 - Cache aliasing?
 - Prefetch issues?
 - Non-temporal prefetch/store issues?
 - Coldstart configuration of memory controller?
 - Something inherit in Catamount?

Dips Due to DRAM Page Conflicts (Bank Conflicts)



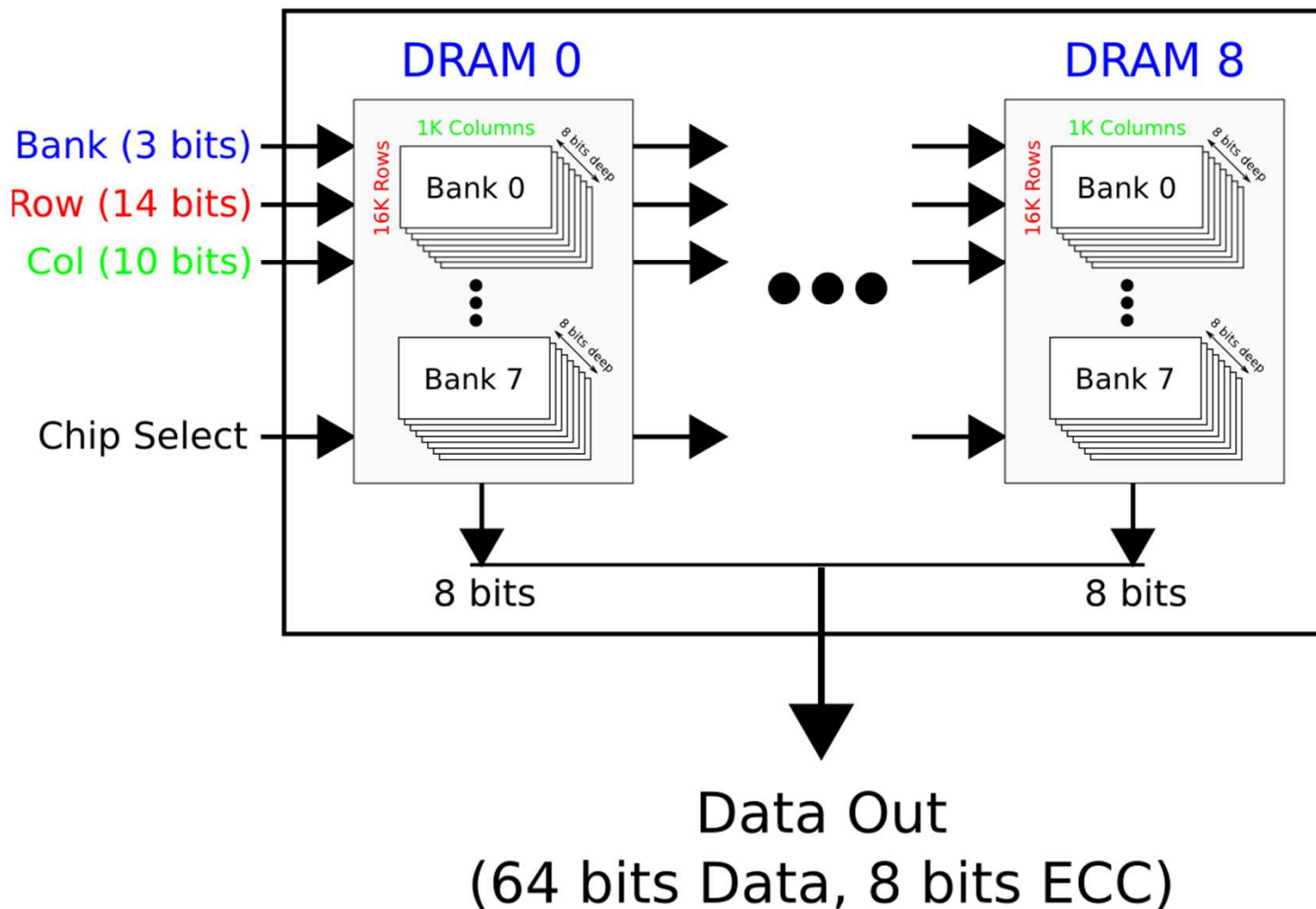


A (Very) Brief DRAM Overview

- Commodity component, most numerous in system
- 2-D array of memory
 - Addressed by (row, column, bank)
 - Accesses to different rows of same bank conflict
 - **Conflicts are slow, prevents request pipelining**
- Typical row (aka page) sizes:
 - DRAM: 1 KB wide (1K columns, each 8-bits deep)
 - DIMM: 8 KB wide (8 DRAM chips in parallel)
- See “Memory Systems: Cache, DRAM, Disk” book

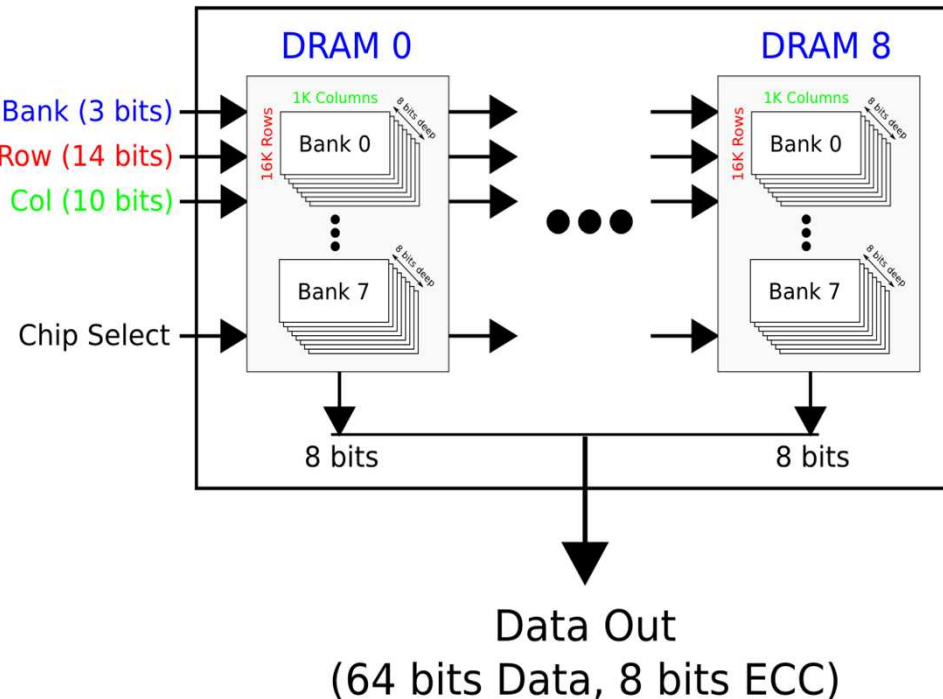
DDR2 DIMM Architecture Example

72-bit Wide DIMM (64-bit Data, 8-bit ECC)



Red Storm DDR2 DIMM Architecture

72-bit Wide DIMM (64-bit Data, 8-bit ECC)



Each DRAM Row is
 $1\text{K columns} \times 8 \text{ bits} = 1\text{K bytes}$

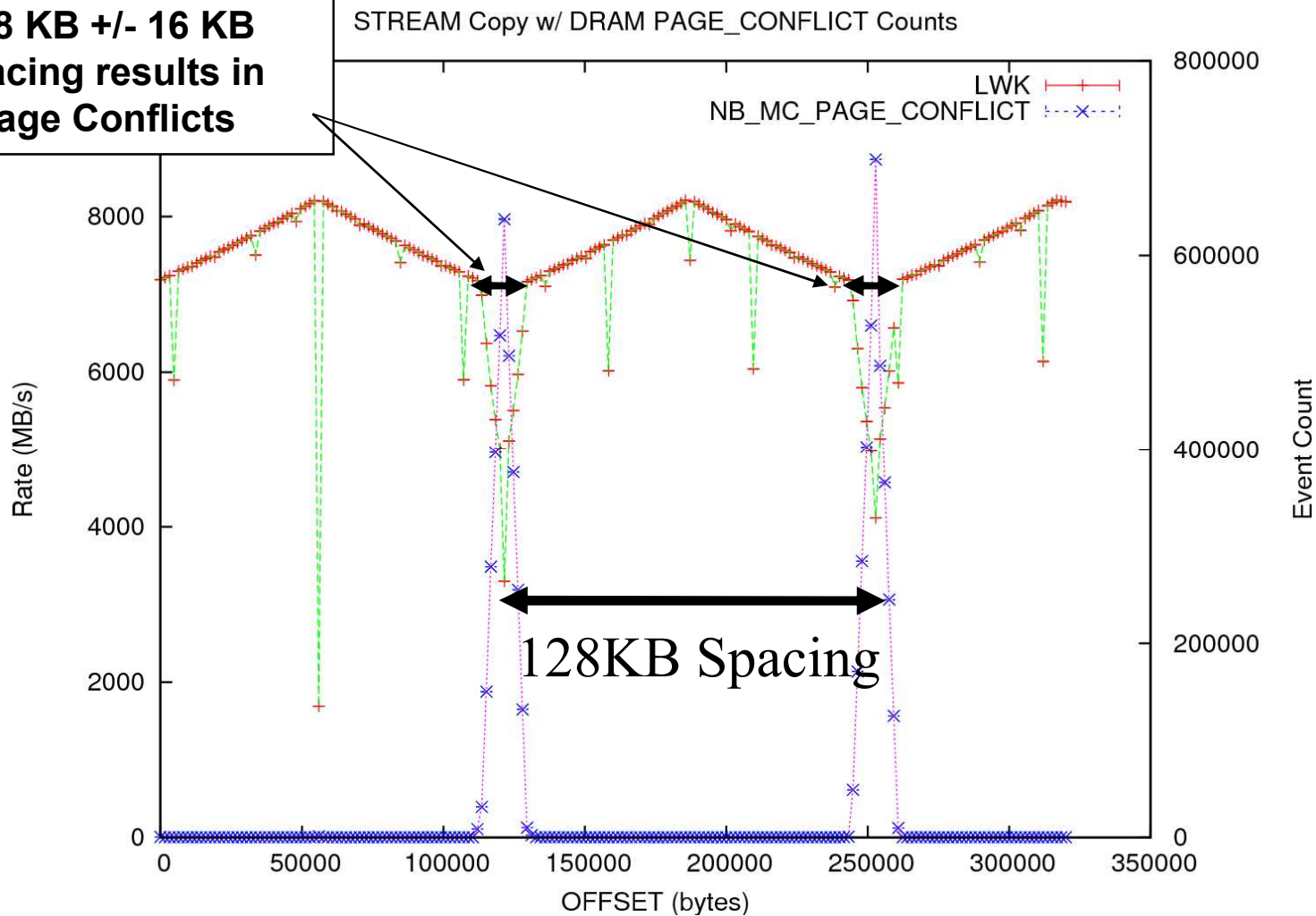
Each DIMM Row is
 $1\text{K bytes} \times 8 \text{ chips} = 8\text{K bytes}$

Each Memory "Page" is
 $8\text{K bytes} \times 2 \text{ DIMMs} = 16\text{K bytes}$

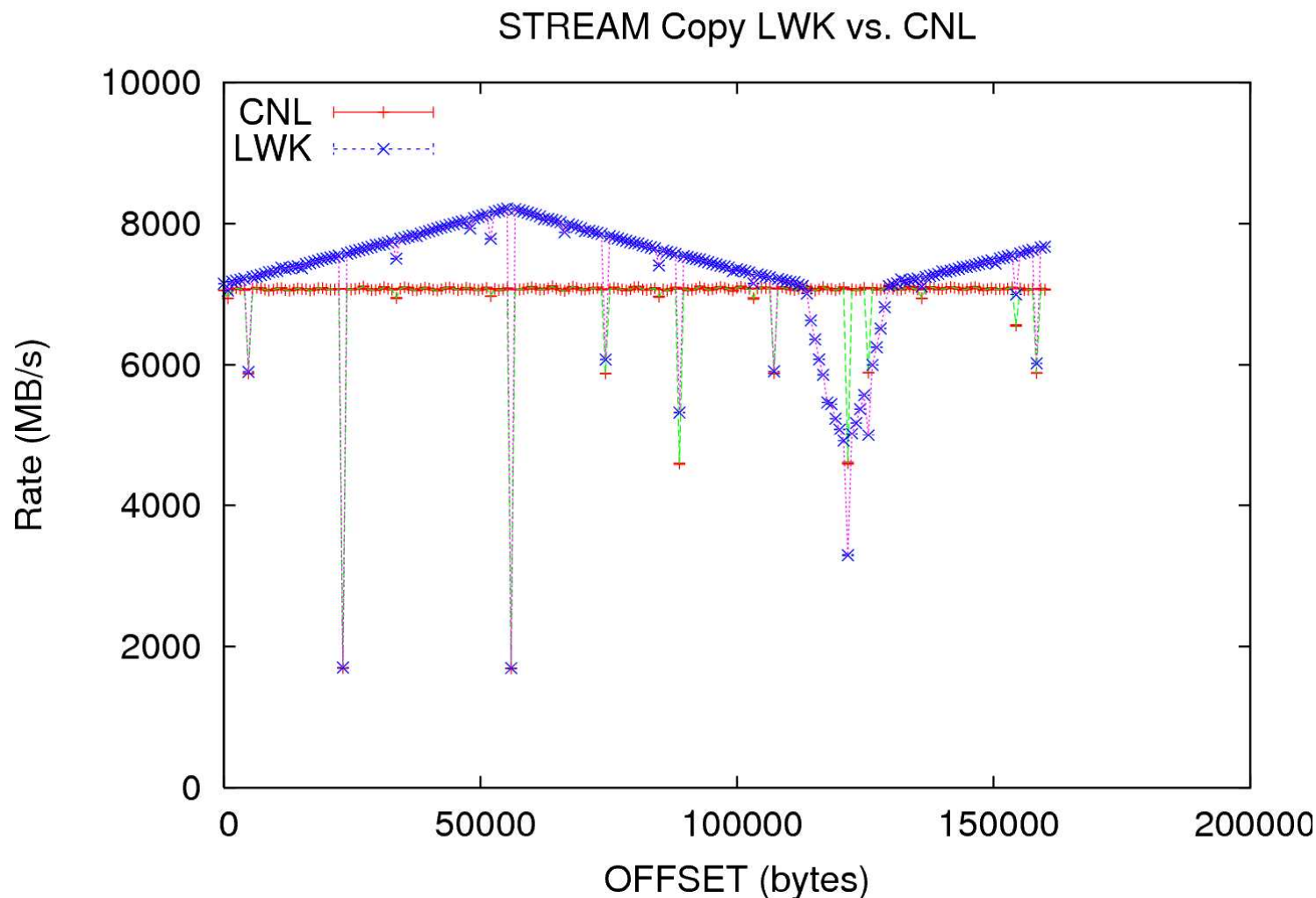
Addresses that are
 $16\text{K bytes} \times 8 \text{ banks} = 128\text{K bytes}$
apart will result in a **Bank Conflict**
(Consecutive accesses to
different rows in same
bank, aka **Page Conflict**)

By the Numbers ...

**128 KB +/- 16 KB
spacing results in
Page Conflicts**



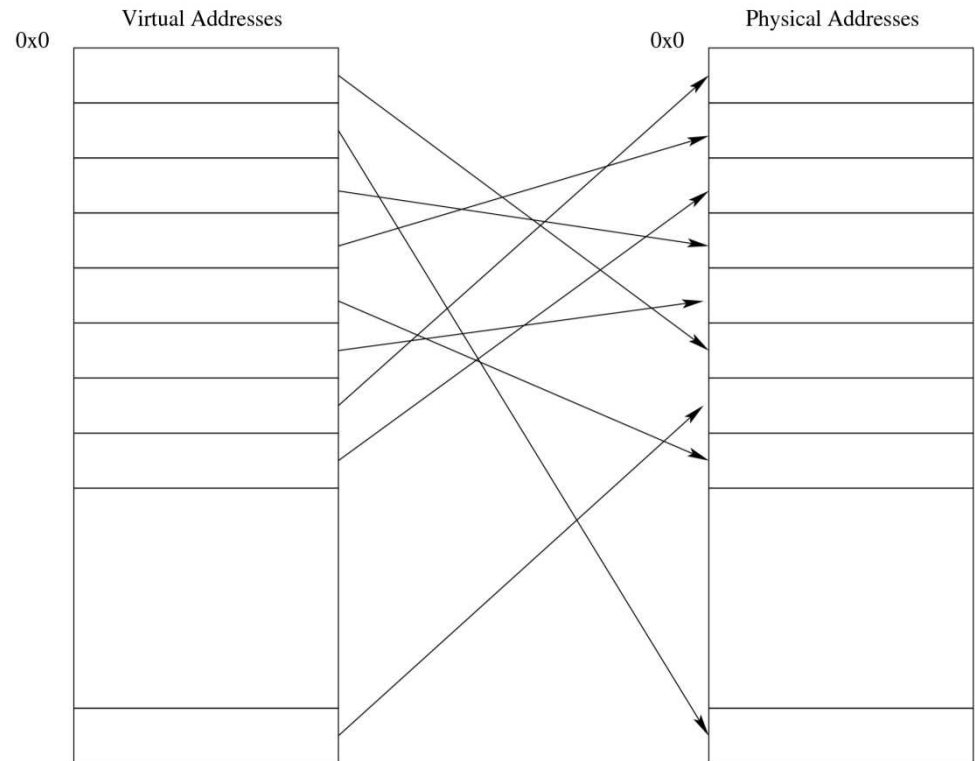
What About Compute Node Linux?





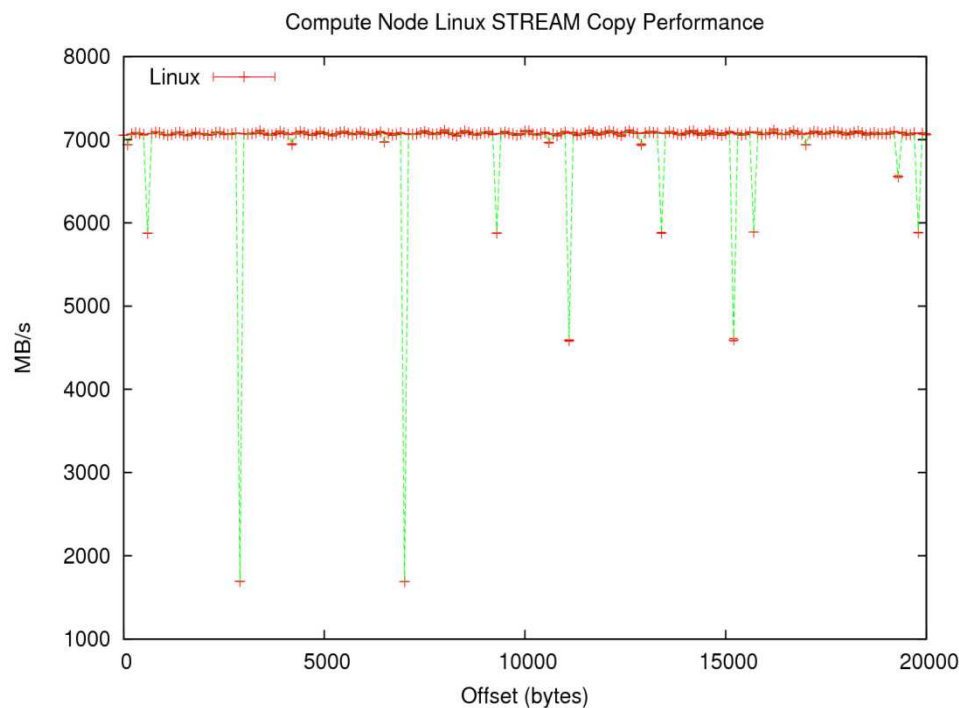
Linux Translation Strategy

- **Will scatter virtual pages throughout the physical space**
- **Mapping is non-deterministic and varies from run-to-run**



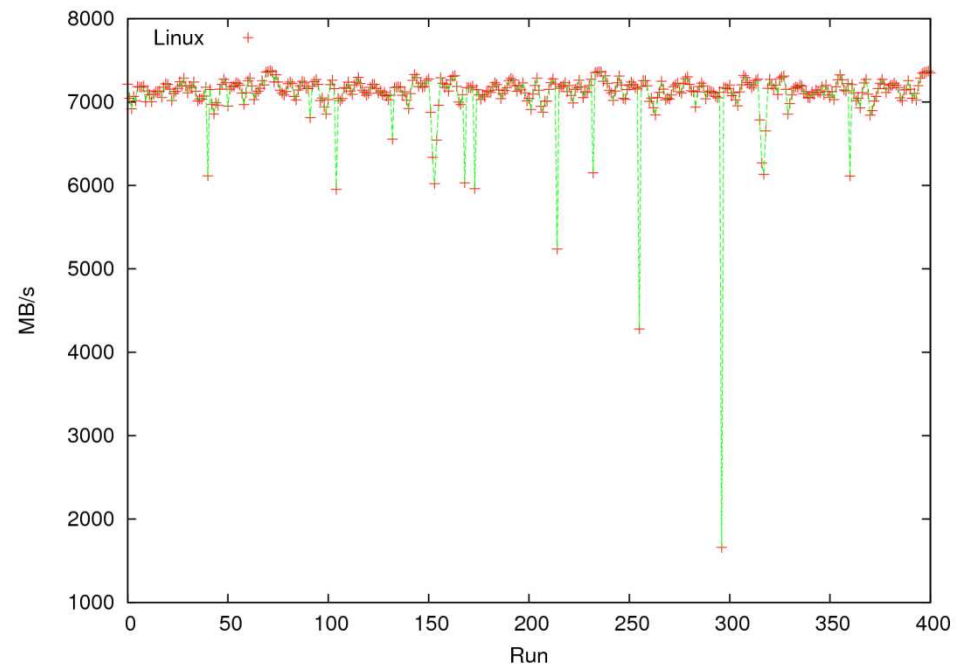
Compute Node Linux Numbers

- Each point from a freshly booted CNL node
- Dips from cache aliasing and also seen on Catamount



As Memory Fragments, Performance Affected

- Translations vary for each application run
- Worst case 80% slowdown due to buffer conflicts and cache aliasing
- Average case similar to best case





Research Questions

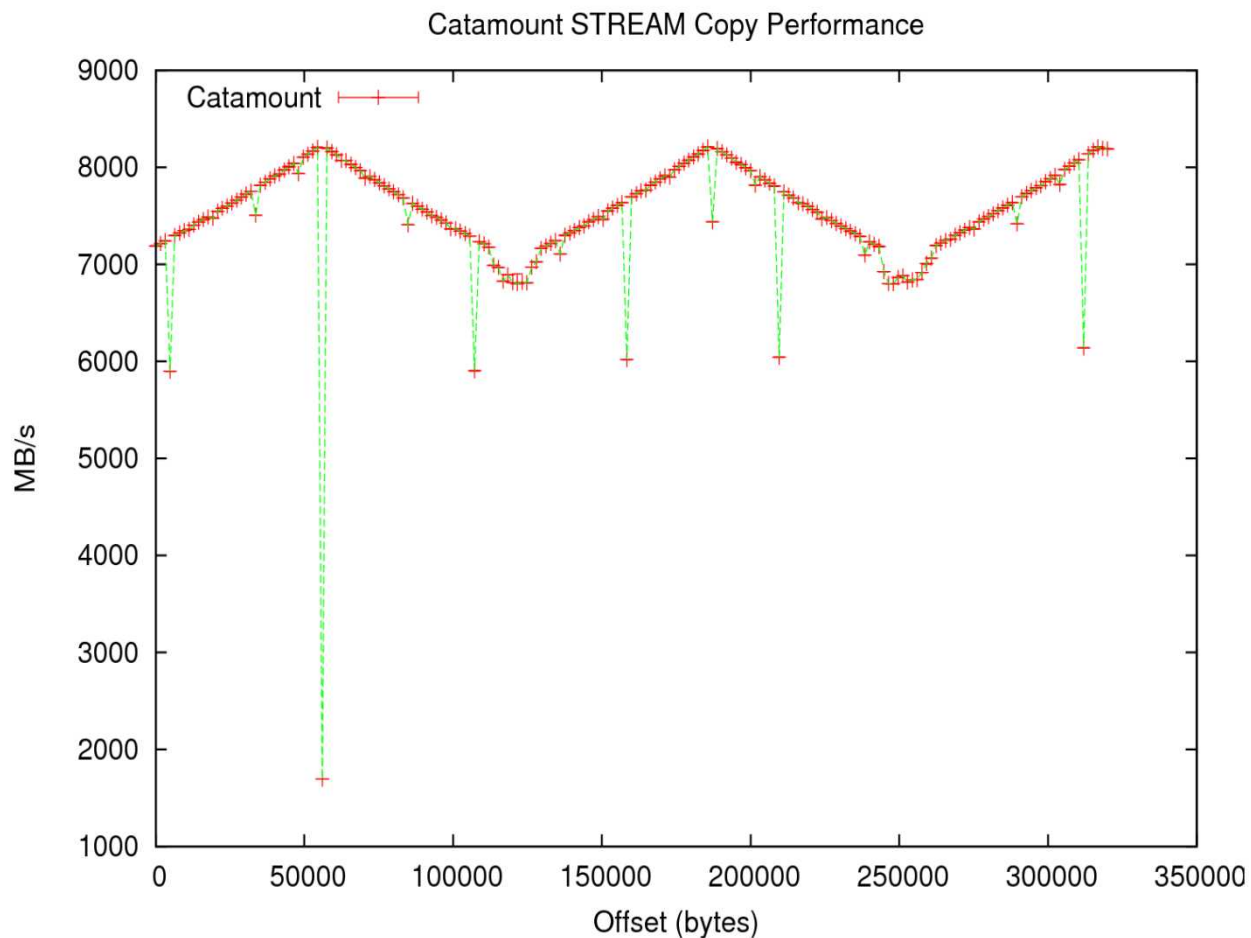
- **Do page conflicts matter for any real applications?**
 - Potential cause of the observed CNL vs. Catamount performance differences on Red Storm?
- **Mitigation techniques:**
 - Opteron memory controller “swizzle” mode
 - Randomize virtual->physical mapping
 - Deterministic virtual->physical mapping
 - No page pinning/unpinning
 - Send address/length to SeaStar vs. command array
 - Compiler optimization?
 - Stream-style programming...
 - 1 array with unit stride cannot cause bank conflict



Adaptive Approaches

- **Monitor page conflict counts while an application runs**
- **If system sees application page conflict counts increasing, shuffle memory mapping**
- **Intension: cap the number of page conflicts at a certain level**

Adaptive Page Mapping Performance



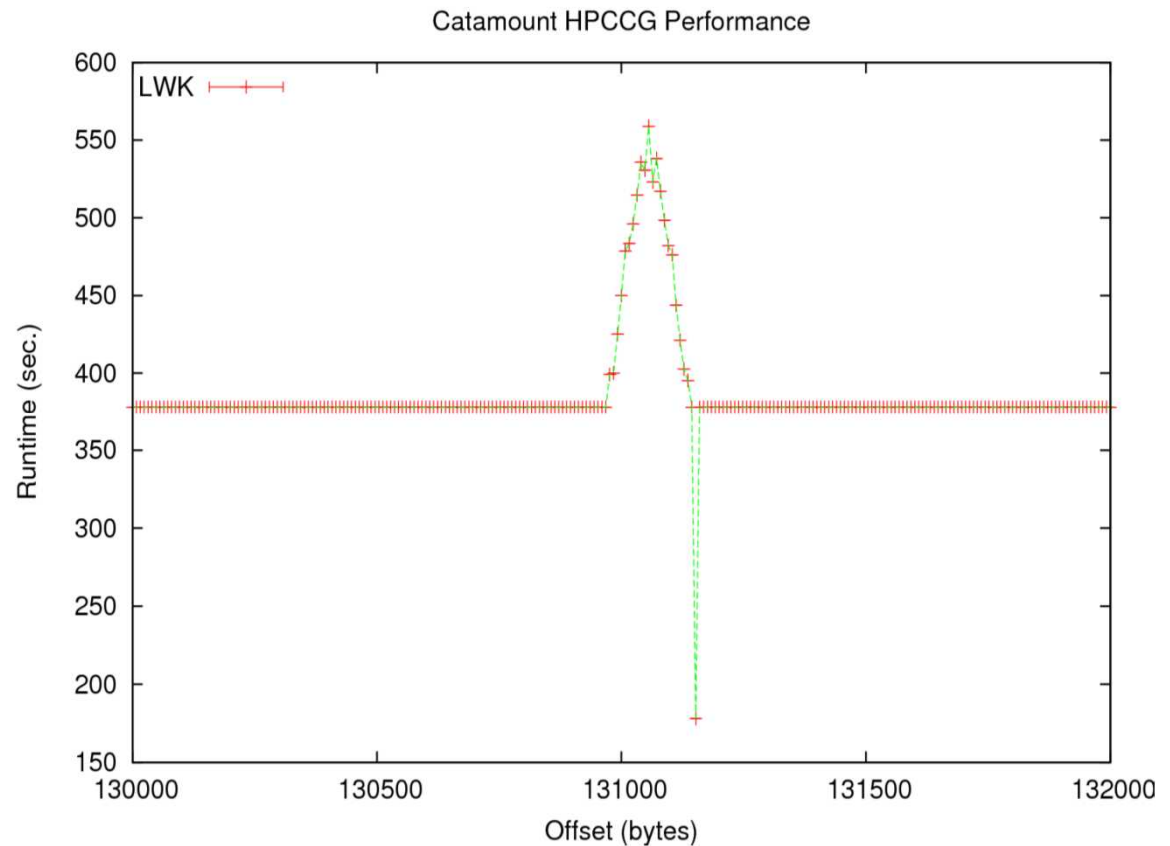


What About Real Applications?

- **HPCCG: somewhere between a micro-benchmark and a real application**
- **Written by Mike Heroux of Sandia National Labs**
- **Simple preconditioned conjugate gradient solver**
- **Generates a 27-point finite difference matrix with a user-prescribed sub-block size on each processor**
- **Processor domains are stacked in the z-dimension**

HPCCG – Page Conflict Slowdown

- 32 nodes
- Offset identical on each node
- ~50% slowdown





Summary

- **Virtual to physical translations can affect the performance of HPC applications**
- **DRAM page buffer is another level of locality in the memory hierarchy that the programmer has little control over and may be important to application performance**
- **No translation strategy clear winner**



Experimental Platform

- **Hardware**
 - 32 node Cray XT3/4 dev system at SNL
 - 2.4 GHz, dual-core AMD Opteron w/ 4 GB RAM
 - Cray SeaStar NIC
- **Software**
 - Catamount lightweight OS
 - Cray Compute Node Linux