SAND2011-4740P

ACES/Cray High Performance Interconnect Project

Scott Hemmert
Scalable Computer Architectures
Sandia National Laboratories
Albuquerque, NM

Sandia National Laboratories is a multi-program laboratory operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin company, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

NNSA National Nuclear Security Administration

Sandia National Laboratories

Project Overview

- NNSA/ASC asked ACES to consider definition of, and technical oversight for D&E project with Cray, Inc.

- Cray Interconnect Genealogy
  - Generation 1: SeaStar, XT architecture interconnect
  - Generation 2: Gemini, XE architecture interconnect
  - Generation 3: Aries, to be deployed with “Cascade” architecture

- Final SOW signed July 2010
  - Will primarily focus on a potential future, post Aries, interconnection network referred to as *Pisces*, tentatively planned for the CY 2015 timeframe

- Early example of focused co-design
  - Efforts centered around impact on ASC applications

High Level Project Tasks

- NIC Studies and Analysis
  - Analyze Gemini interconnect performance
  - Look for improvements which can be enhanced in Pisces
  - Focus on NIC, with emphasis on occupancy, latency, MPI message throughput, and independent progress.

- Router & Network Studies and Analysis – Analyze Aries interconnect performance
  - (byproduct) Optimize Aries network settings for ASC applications
  - Look for possible enhancements to Pisces – Focus on network routing

- Pisces – Initial architectural specification in collaboration with ACES
  - Perform a comparative study between Pisces and other state of the art interconnects, such as InfiniBand, and possibly one or two others

Challenge Areas for HPC Networks

- The traditional “big three”
  - Bandwidth
  - Latency
  - Message Rate (Throughput)

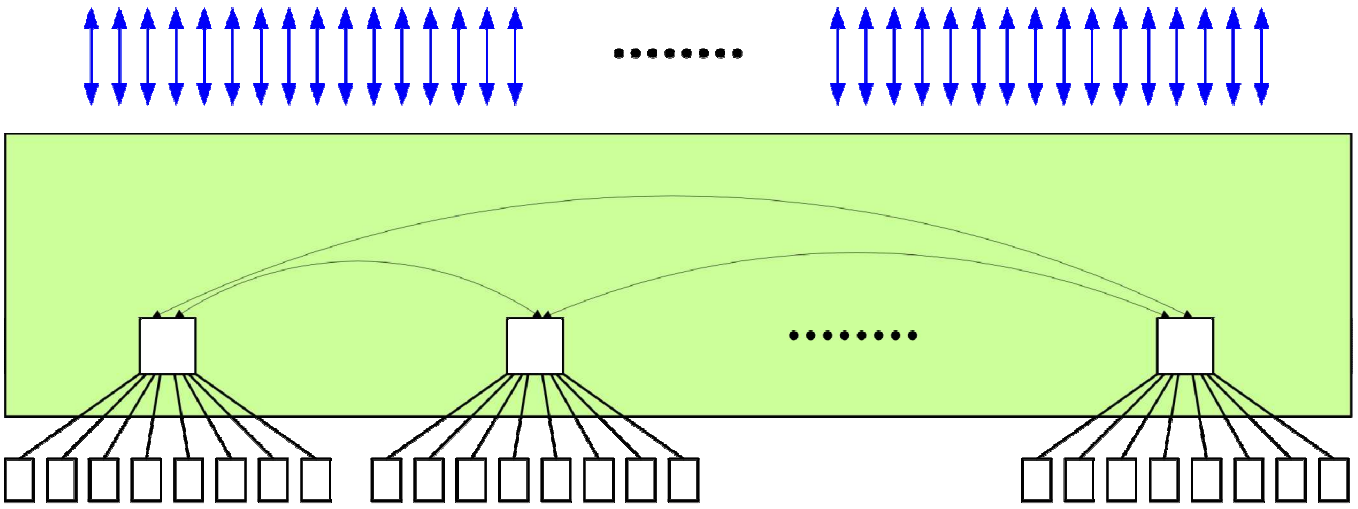
- Other important areas for “real applications” versus benchmarks
  - Allowable Outstanding Messages
  - Host memory bandwidth usage
  - Size of required buffering
  - Noise (threading, cache effects)
  - RDMA effects
  - Topology
  - Reliability

Early Analysis of Aries Routing using ASC Apps

- Aries implements a Dragonfly topology
  - Form of hierarchical all-to-all topology
  - Overview to follow
- Initial traces for CTH and Sage used
  - CTH trace uses 1024 MPI ranks
  - Sage trace uses 8192 MPI ranks
- Following slides graciously provided by Cray

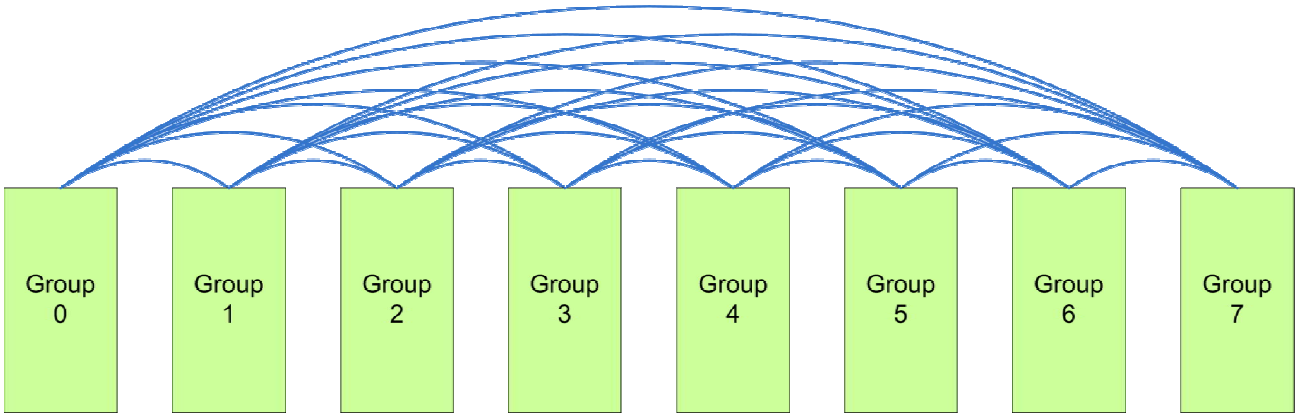
High radix networks – Dragonfly

- Variation on the flattened butterfly
- Construct groups of locally-connected nodes
- Treat the group as single “super node” with very high radix
- Pool all the optical links coming out of the group into a single dimension
- Create a single all-to-all optical stage among groups



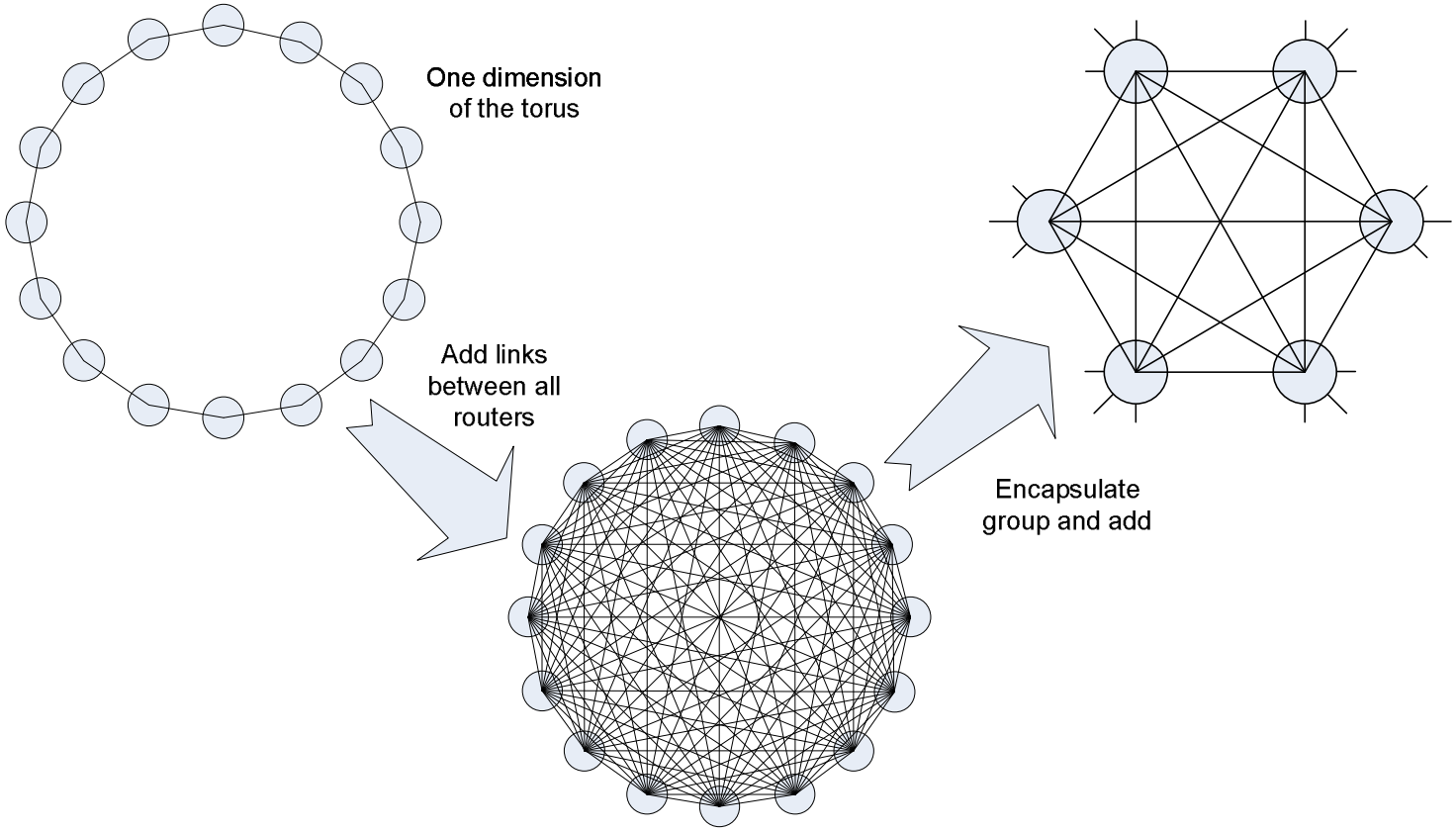
High radix networks – Dragonfly

- Group size is determined by router radix, packaging and electrical characteristics
- Uses extra (inexpensive) local hops to reduce (expensive) global hops
- Very large systems with only a single optical hop for well-balanced traffic
- Bi-section bandwidth scales with system size



$$bisection\ bandwidth = link\ bandwidth \times \frac{group\ optical\ ports}{groups - 1} \times groups$$

Dragonfly from a torus perspective



Routing Nomenclature

- Minimal Routing
  - Shortest path between any two network end-points
  - Works well for global traffic
    - All-to-all
    - Uniform random

- Non-minimal Routing
  - “Randomly” choose intermediate router
    - Route minimally to intermediate router
    - then, route minimally to destination
  - Non-minimal traffic spread using deterministic hashing function
  - Good option for deterministic/fixed traffic patterns

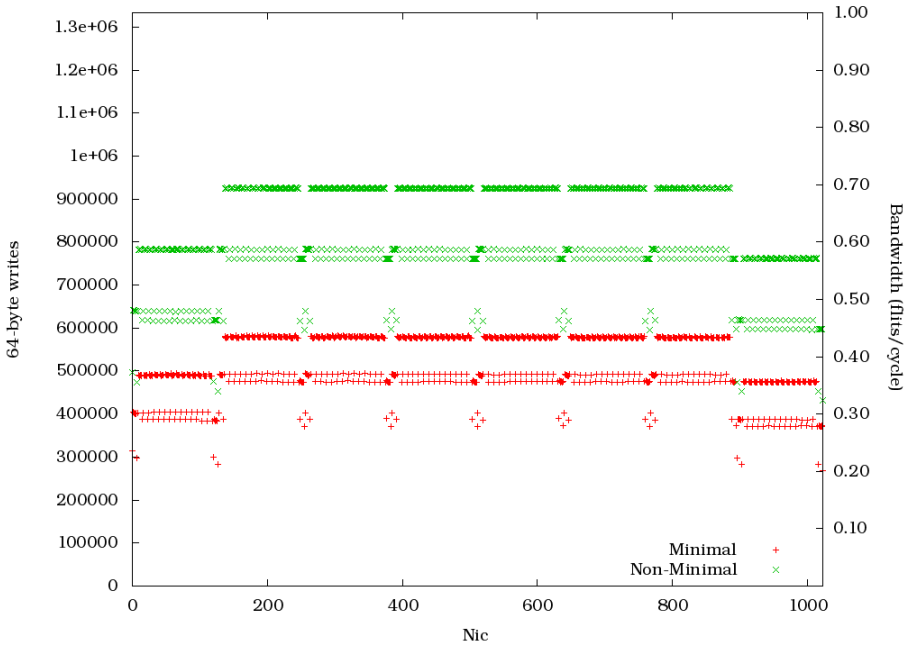
- Adaptive Routing
  - Route chosen based on network load rather than only using a deterministic hash

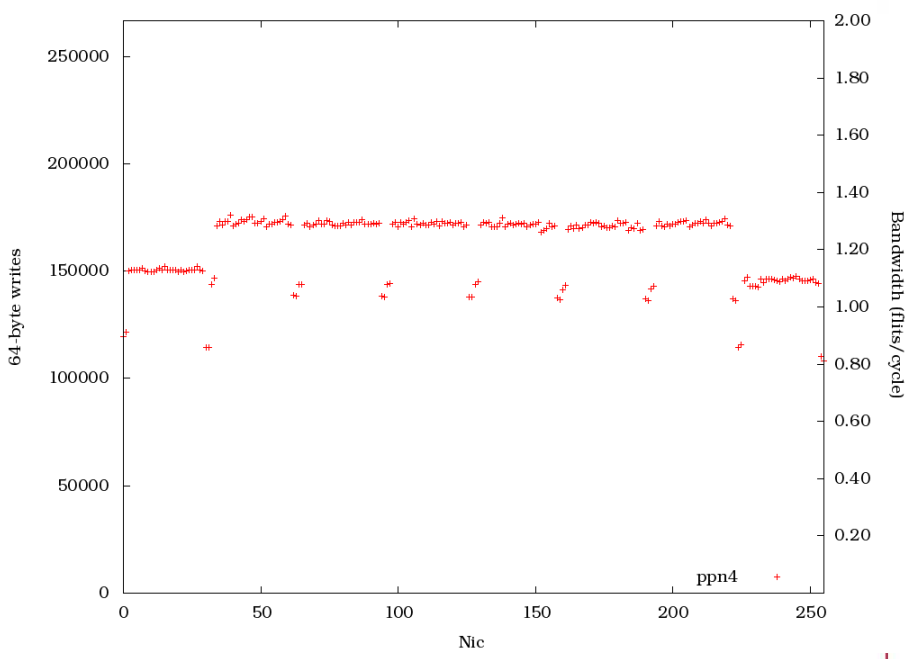
Background on simulator

- Cray in-house network simulator (rtrsim)
  - Cycle accurate simulation of the Aries router
- Dragonfly structure specialised to Cascade
  - Wiring
  - Route tables
  - Adaptive routing algorithms
- NIC model supports either
  - Synthetic traffic generator
  - Multiple MPI ranks, with a trace per rank
- Parallel application with each MPI task manages a number of routers
  - Simulations scale well, typical runs use 192 or 384 tasks.

CTH – Initial results

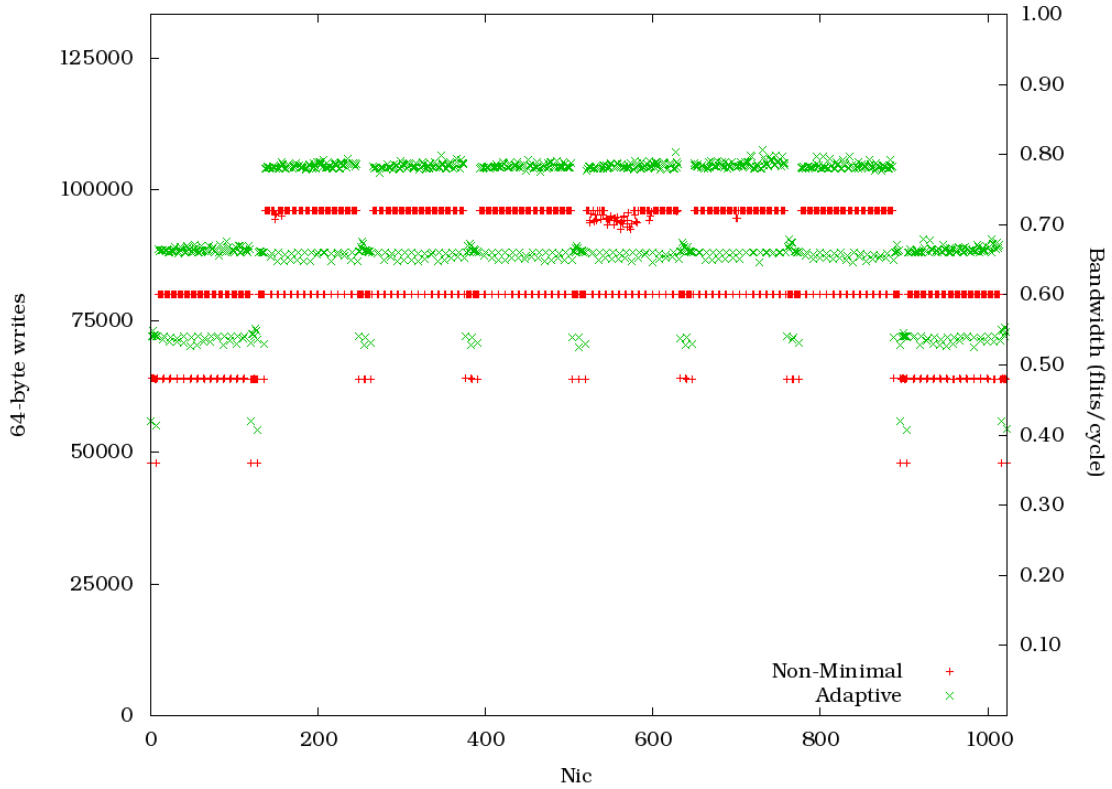
- Comparison of minimal and non-minimal routing
- Significant increase in bandwidth with multiple (4) traces per NIC





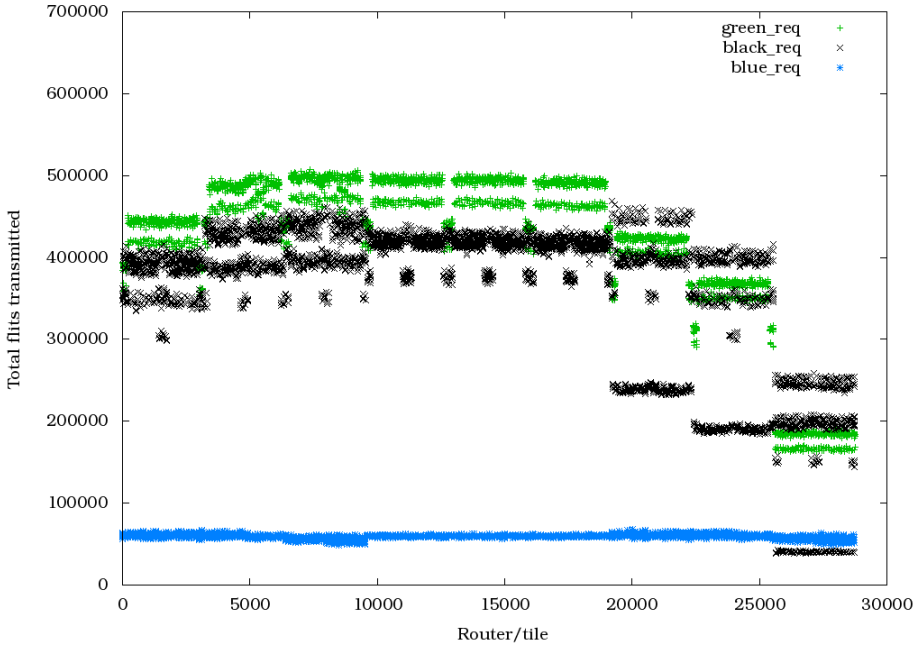
CTH – Adaptive Routing

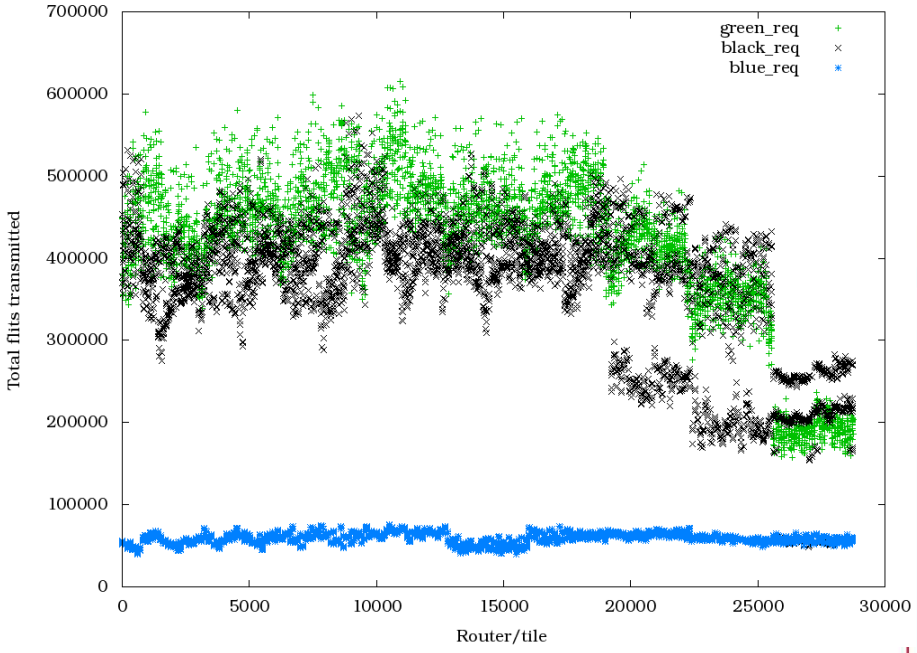
- Significant improvements in bandwidth with adaptive routing



CTH Results – Link Loading

- Comparison of non-minimal and adaptive routing
- Both show low loading on the global links

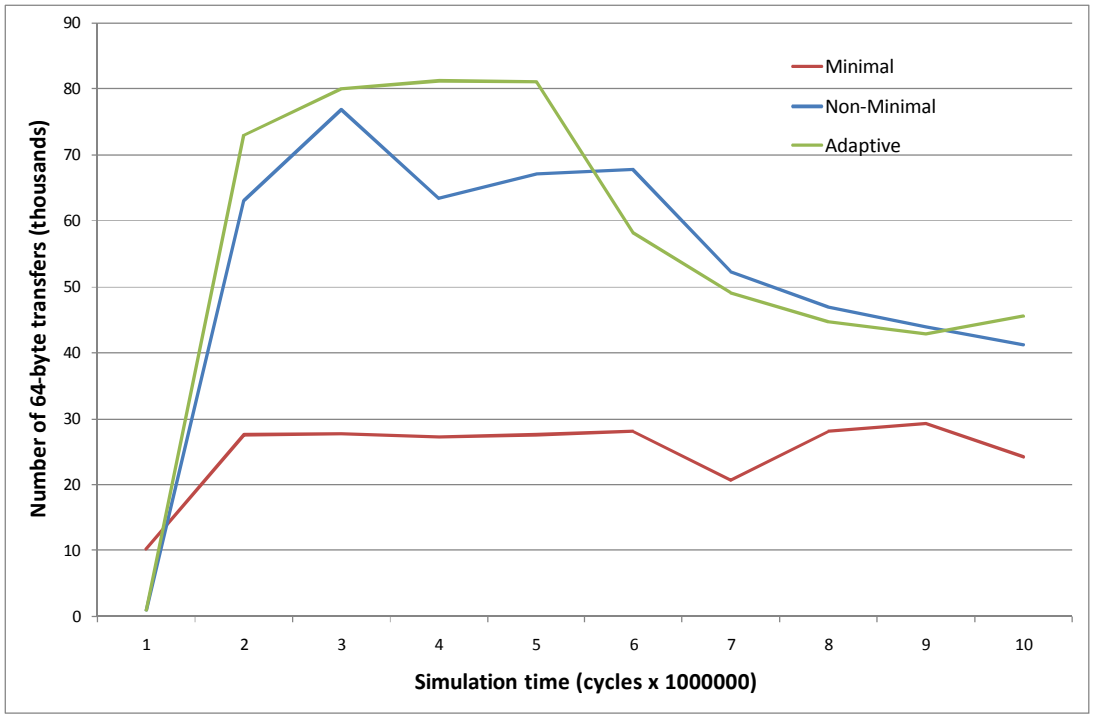




Supplied by Cray, Inc.

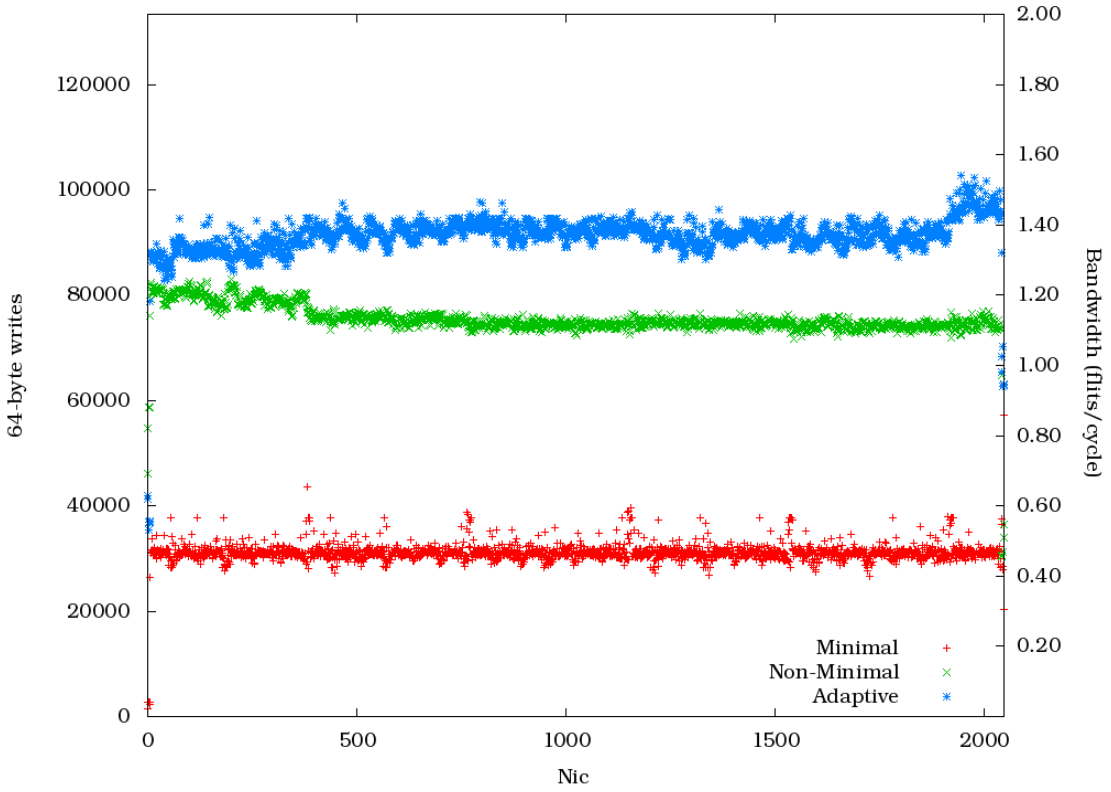
Sage – Initial Results

- Big variations in bandwidth across one cycle of the trace



Sage Results

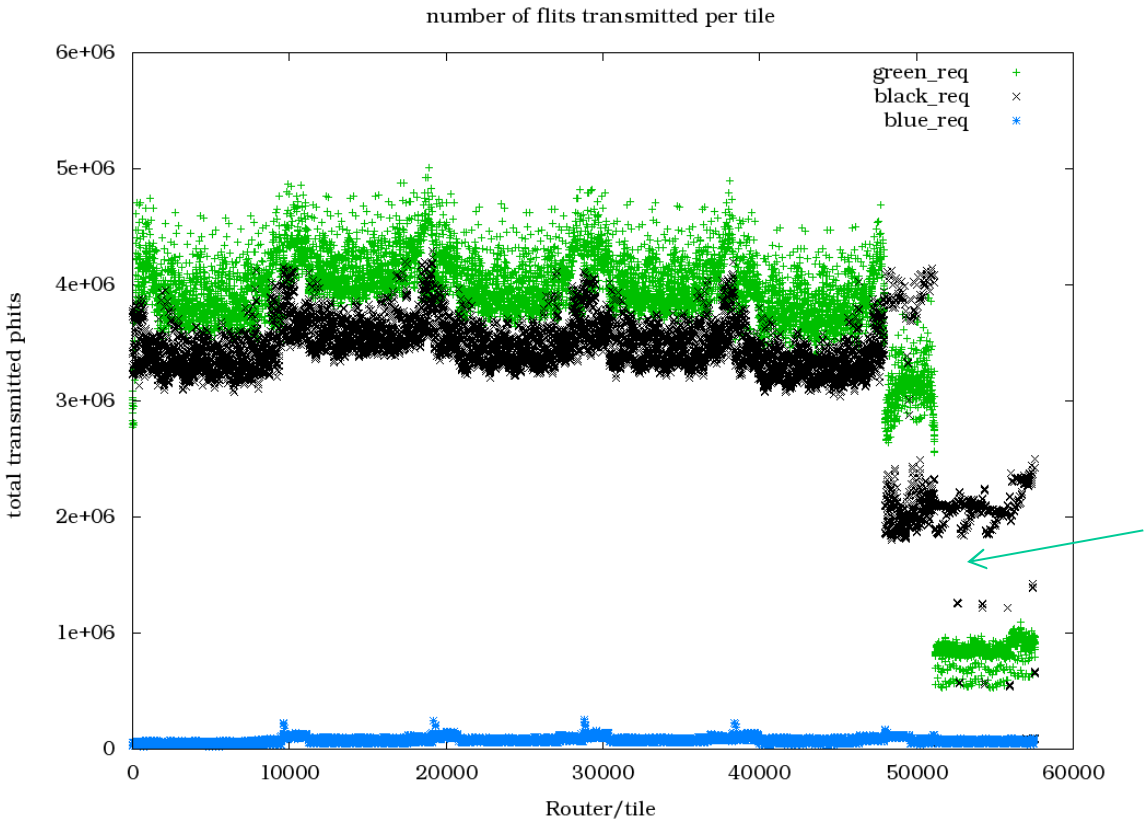
- Significant benefits from adaptive routing



Supplied by Cray, Inc.

Sage Results – Link Loading

- Low loading on the global links



Conclusions from Simulation Work so far

- High percentage of peak bandwidth obtained
  - Trace based results include effects of collectives and load imbalance
  - With just point-to-point traffic bandwidths are close to peak
- Significant benefits from adaptive routing
- CTH and Sage show low global link loading
  - 50% global bandwidth is plenty
- Next Step
  - Look at AMR step in Xnobel
- From Cray Perspective
  - A successful demonstration of the benefits of CoDesign approach.

Future Routing Studies

- More applications
  - Need applications with different communication patterns
  - Would like representative applications from all 3 NNSA labs
- Larger node counts
  - Would like to be able to have more traces per NIC
  - May be limited by simulation runtimes
- Multiple simultaneous applications
  - Determine if applications will interfere with each other

Next Step: NIC Architecture

- Focus on:
  - Latency
  - MPI message throughput
  - Independent progress
  - Processor Occupancy

- Wide range of possibilities
  - Full MPI offload
  - Full MPI onload
  - Using “spare” cores for MPI processing
    - What’s done on dedicated cores?
    - What’s done on application cores?
    - Coherency issues
    - Memory performance issues
    - MPI Performance issues (serializing work through fewer cores)

Questions