High Speed Asynchronous Data Transfers on the Cray XT3

Ciprian Docan, Manish Parashar and Scott Klasky
The Applied Software System Laboratory
Rutgers, The State University of New Jersey

CUG 2007, Seattle, WA
May, 2007

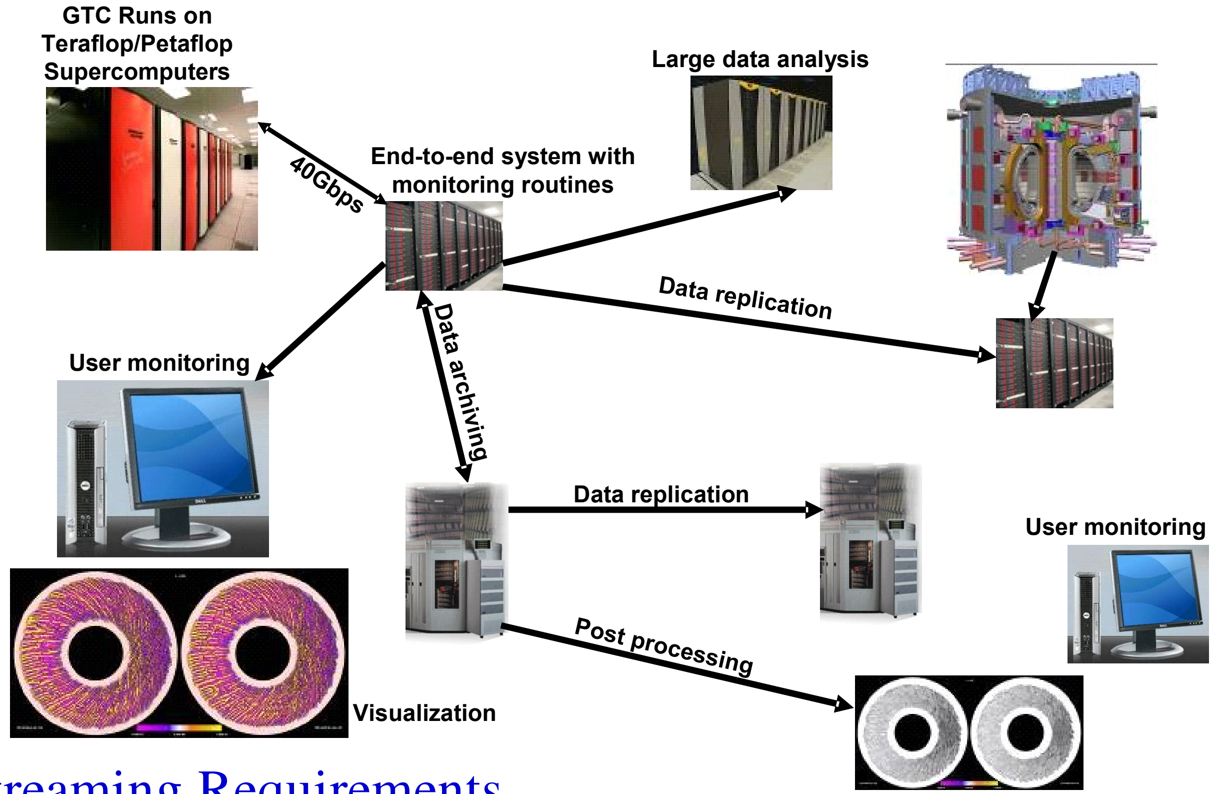
T The
A Applied
S Software
S Systems
L Laboratory

Outline

- Data Streaming Requirements in the Fusion Simulation Project
  - Support for Code Coupling & Asynchronous Data Movement
- DART Overview
- DART Architecture and Description
- DART Evaluation
- Conclusions and Future Work

TASSL: The Applied Software Systems Laboratory

Data Streaming Requirements in the Fusion Simulation Project (FSP)



- Data Streaming Requirements
  - Enable high-throughput, low latency data transfer to support near real-time access to the data
  - Minimize related overhead on the executing simulation
  - Adapt to network conditions to maintain desired QoS
  - Handle network failures while eliminating data loss

The Applied Software Systems Laboratory

Support for Code Coupling & Asynchronous Data Movement

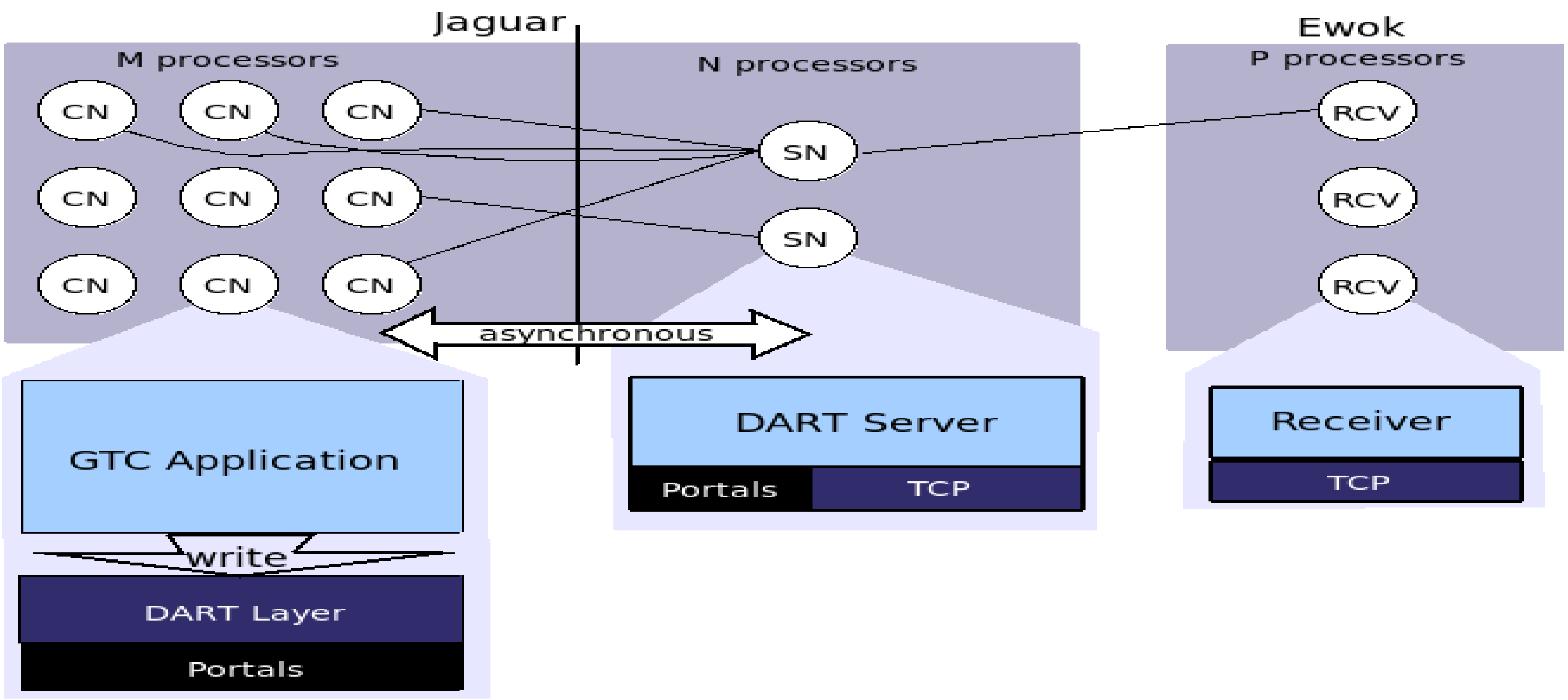
- Seine: Dynamic, semantically-specialized shared-spaces for code coupling
  - High-level (shared-space) programming abstractions, efficient and scalable runtime
- DART: Light-weight substrate for asynchronous, low-overhead/high-throughput data IO on petascale system
  - Based on Portals and RDMA
- ADAPT: Middleware for autonomic wide-area data streaming and in-transit data processing
  - Enable high-throughput, low latency data transfer to support near real-time access to the data
  - Effectively outsource processing to in-transit processing nodes
  - Minimize related overhead on the simulation
  - Adapt to network conditions to maintain desired QoS

DART: A Substrate for Asynchronous IO

- Objectives: Alleviate impact of IO on scientific simulations
  - Minimize total IO time on compute nodes
  - Maximize data throughput on compute nodes
  - Minimize overhead on data transfers (packet header size)
  - Minimize IO computational overhead on compute nodes
    - Data transfer logic, data buffering
- Key idea – Overlap transfers with computations

- DART Client – Runs on compute nodes
  - Lightweight & Simple!
    - maintains buffers for data transfers
    - notifies server when data is available
- DART Server – Runs on service/IO nodes
  - Contains logic and buffers
    - pulls data from compute nodes on notification
    - performs requested IO (i.e. file system, socket, etc.)
- Data transfers are asynchronous and decoupled

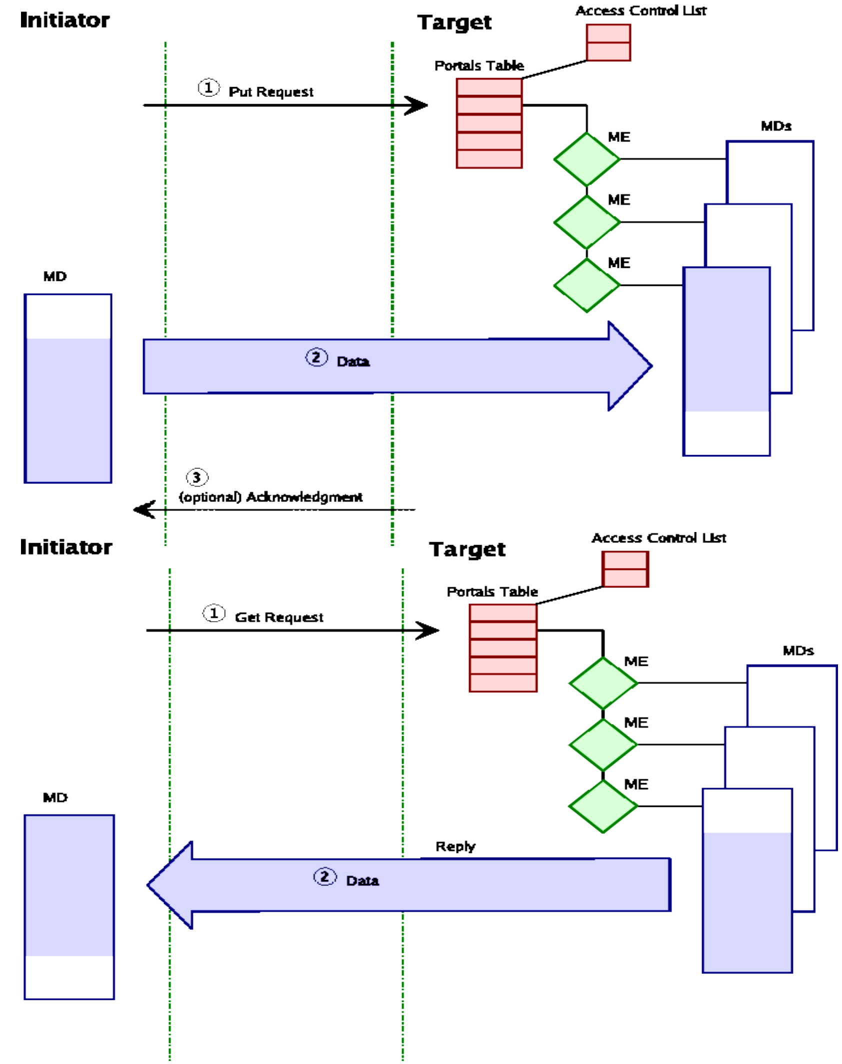
DART: Architecture



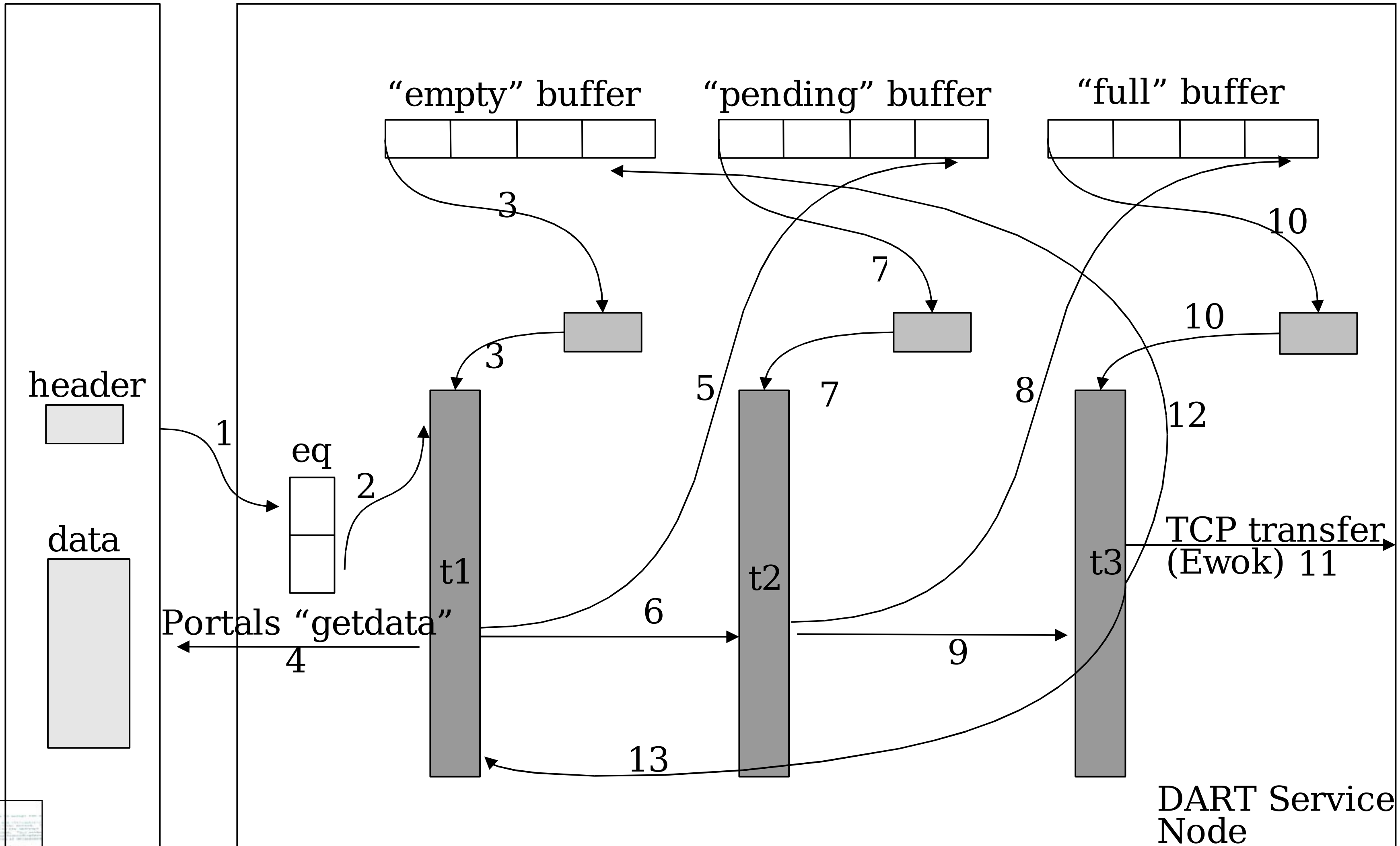
- Compute and Service nodes communicate using the Portals library
- Service nodes and receivers on remote cluster (i.e. Ewok) communicate using TCP sockets

The Portals Library

- RDMA implementation with OS and Application bypass
- Put operation
  - Initiator pushes its MD content into the target's address space
  - Events are recorded
- Get operation
  - Initiator fetches the target's MD content into its address space
  - Events are recorded

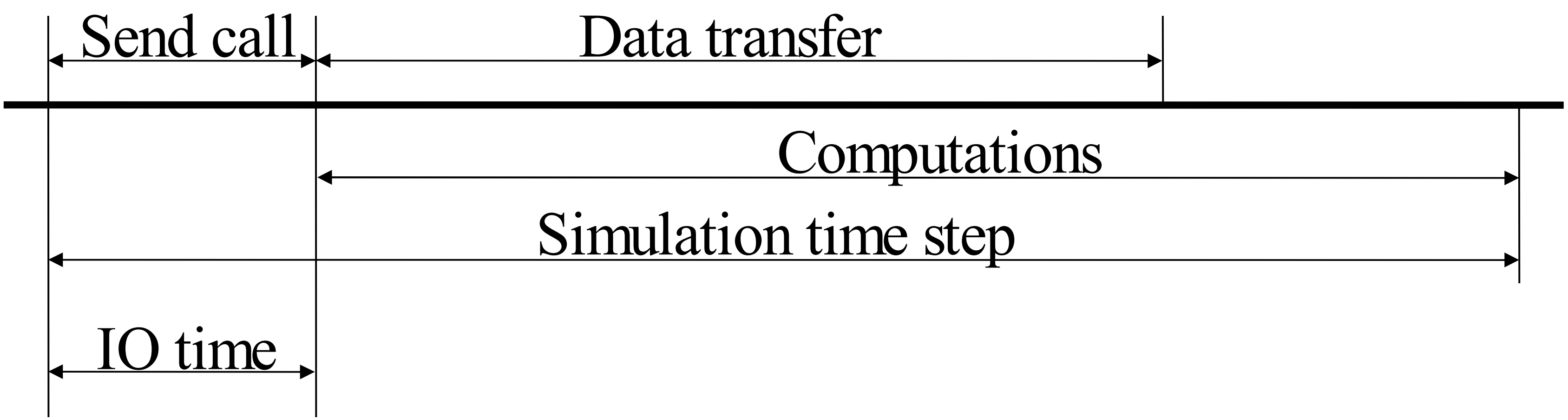


DART: Server Operation



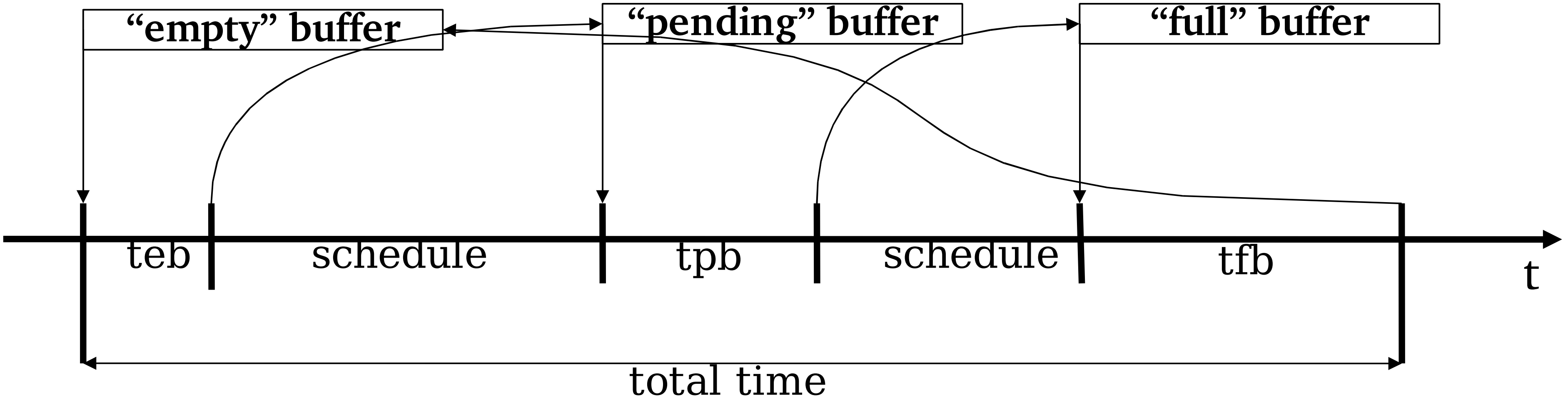
DART: Time Sequence at the Client

- DART asynchronous IO calls at the client
  - Send calls return before data transfer completes
  - Data transfers overlap with computations
  - IO calls can block if buffer is full or busy



DART: Time Sequence at the Server

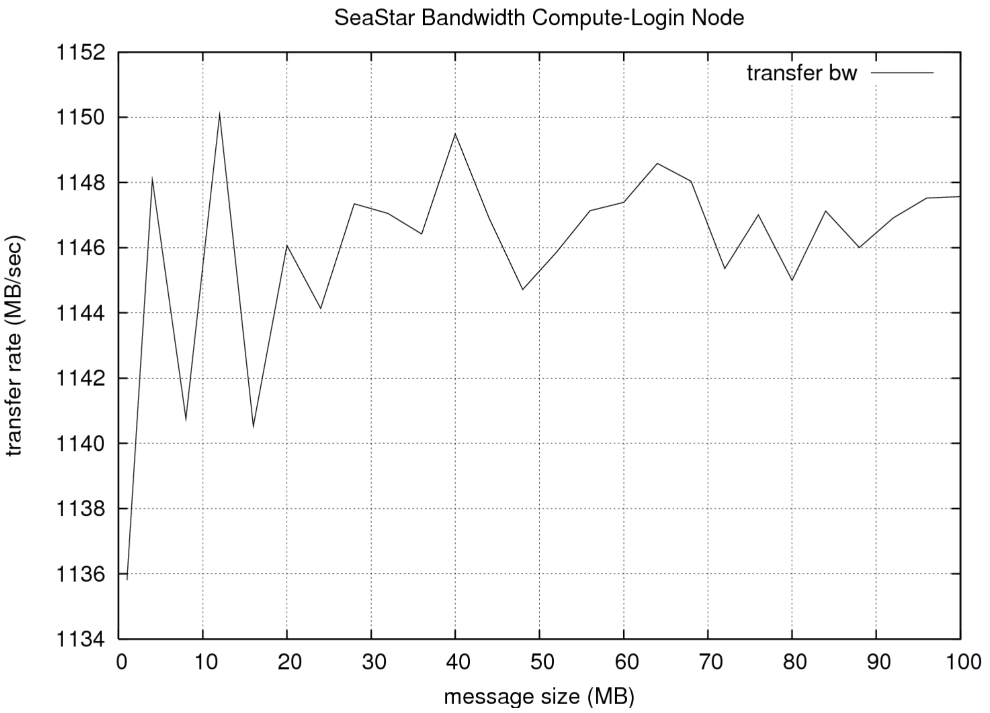
- Asynchronous data transfers
  - *teb* – time spent in the "empty buffer" queue
  - *tpb* – time spent in the "pending buffer" queue
  - *tfb* – time spent in the "full buffer" queue



The Applied Software Systems Laboratory

DART Evaluation: Achieved Transfer Rates

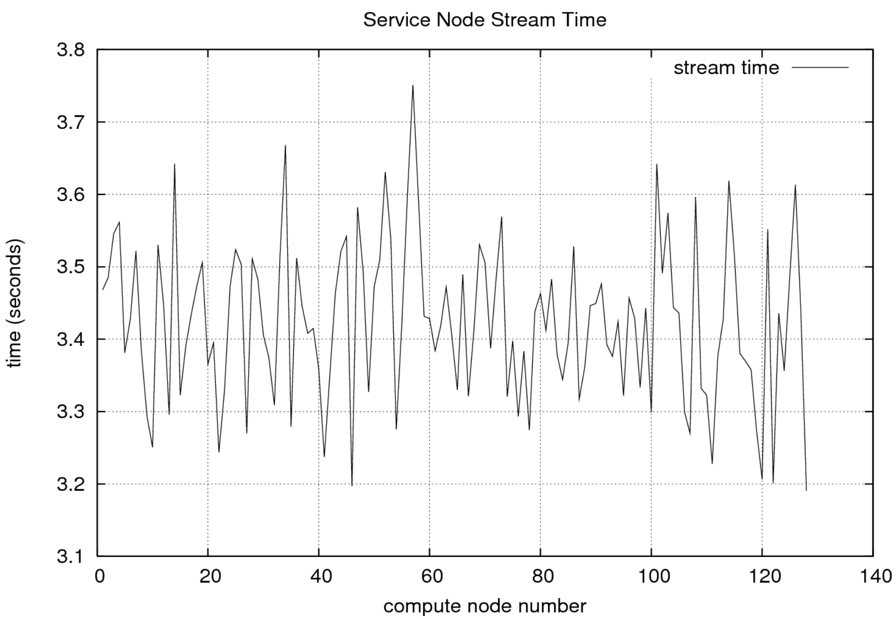
- Maximum transfer rate achieved between CN and SN on Jaguar
  - One DART server and two DART clients
  - Message sizes varied from 1 MB to 100 MB using 4 MB increments
  - Data dropped at the service node

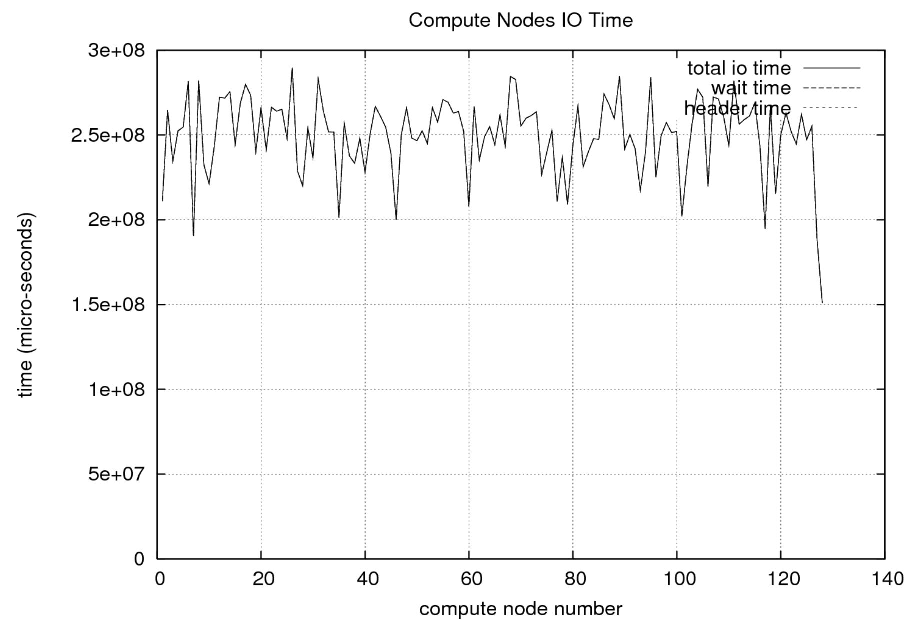


The Applied Software Systems Laboratory

DART Evaluation: Influence of Compute Time on IO Overhead

- Stream time on service node
  - Data sends are sequential ~ 3.4 sec cumulative time over 100 iterations
- Compute time 1 microsecond
  - No overlap – IO operation sequential



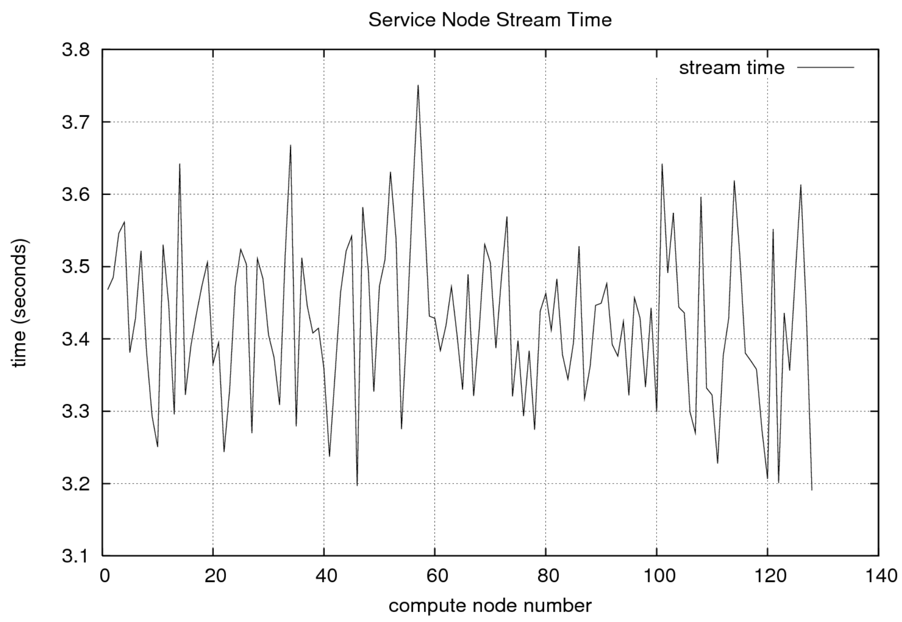


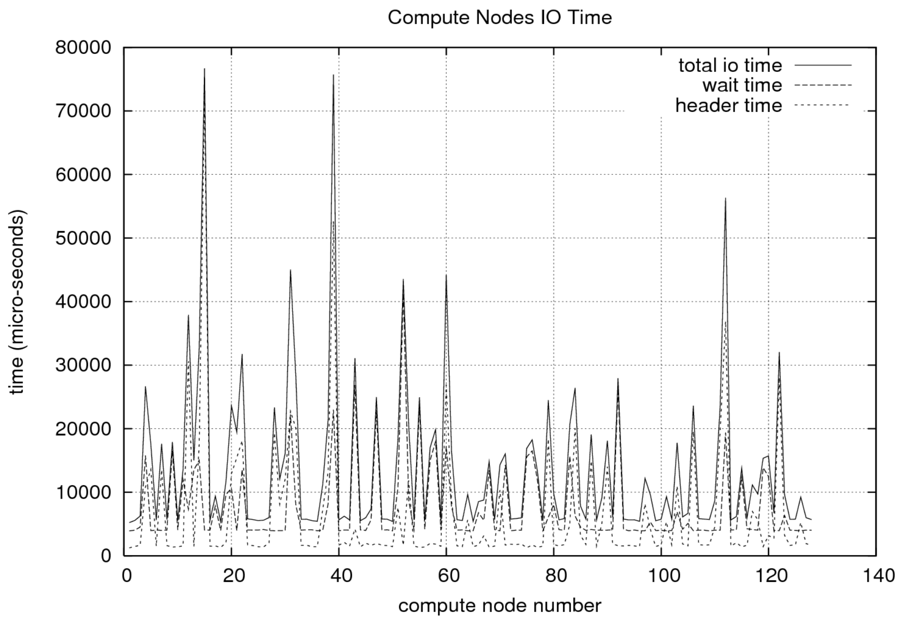
- Optimum compute time value: # CN * stream time

The Applied Software Systems Laboratory

DART Evaluation: Influence of Compute Time on IO Overhead

- Stream time on service node
  - Data sends are sequential ~ 3.4 sec cumulative time over 100 iterations
- Compute time 4.3 sec
  - Maximum overlap – IO operations parallel



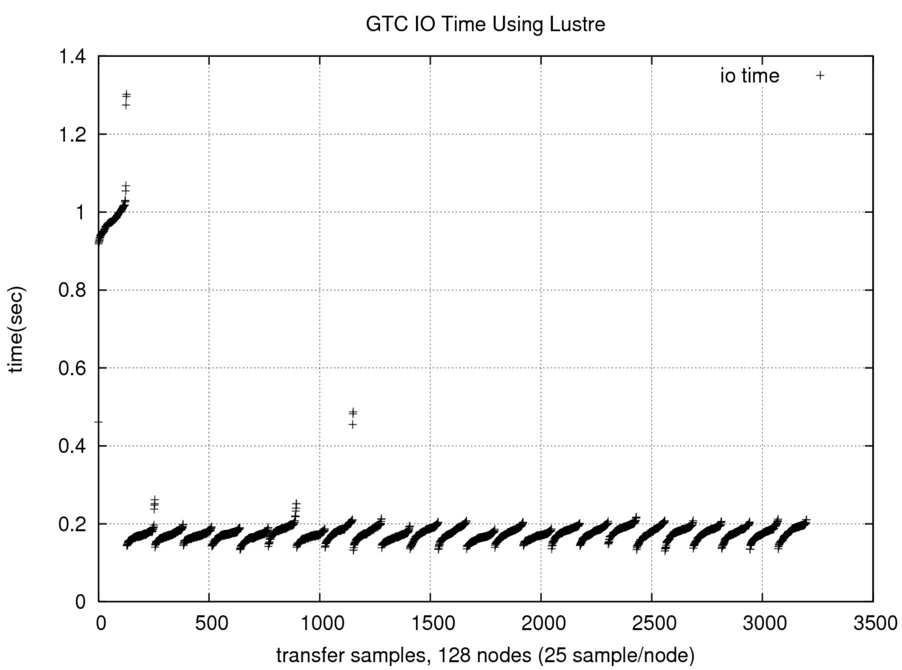


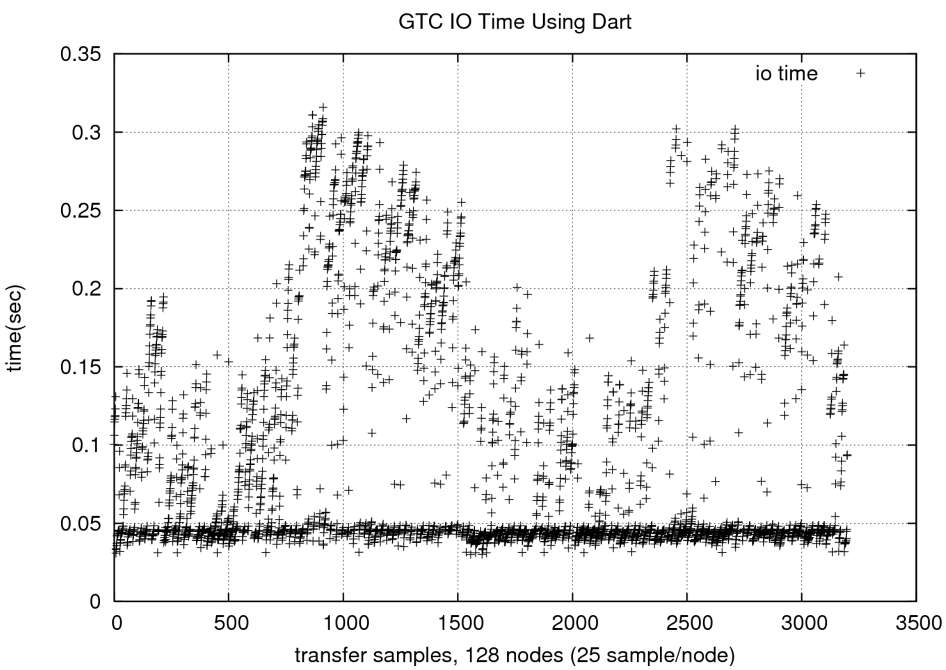
- Need to balance compute time with IO time for maximum overlap

DART Evaluation: Tests Using the GTC Application

- Experiment using the GTC fusion application
  - 128 compute nodes served by 1 service node
  - A ~8.9 MB restart file/compute node written every ~4.5 sec
  - 100 simulation steps
- DART write time: ~0.12 sec
- Effective transfer rate ~593Mbps / compute node
- IO overhead on compute node 2.7%

DART Evaluation: Tests Using the GTC Application





Conclusions and Future Work

- High throughput, low latency data streaming is critical for emerging scientific applications
- DART supports low overhead asynchronous IO on the Cray XT3
- DART Performance
  - Maximum achievable transfer rates using Portals
    - Two clients and one server
  - Overhead of IO operations on GTC application
    - Restart files ~ 8.9 MB/node
- Future work
  - Extend DART to support large message sizes
    - Integrate with GTC to enable ~160 MB/node restart files
  - Enable N x M x P coupling on top of DART

TASSL The Applied Software Systems Laboratory

Questions

Thank You !