



---

# Evaluation of an Eager Protocol Optimization for MPI

**Ron Brightwell and Keith Underwood**

**Center for Computation, Computers, Information, and  
Mathematics**

**Sandia National Laboratories  
Albuquerque, New Mexico, USA**



Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,  
for the United States Department of Energy under contract DE-AC04-94AL85000.





# Introduction

---

- **Nearly every MPI implementation uses a rendezvous protocol for long messages**
  - **Optimized for bandwidth**
  - **Mandated by**
    - **Network**
      - **Message selection for appropriate remote destination**
    - **Operating system**
      - **Resource management (pinning pages)**
- **No such restrictions on ASCI Red**
  - **No packetization**
  - **No page pinning**
- **Does an eager protocol make a difference?**
- **Do we need an eager protocol for Red Storm?**



# ASCI Red

- **Hardware**
  - 4640 compute nodes
    - Dual 333 MHz Pentium II Xeons
    - 256 MB RAM
  - 800 MB/sec bi-directional network
  - 38x32x2 mesh topology
- **Software**
  - Puma/Cougar LWK
  - Portals 2.0
- **2.38/3.21 TFLOPS**
- **Deployed in 1997**





# MPI Implementation for ASCI Red

---

- **Short messages (<8KB) are sent eagerly and buffered at the receiver**
- **Long messages are sent eagerly**
  - **Expected messages (pre-posted) are delivered directly into the user buffer**
  - **Unexpected messages leave a message header that allows the receiver to get message from the sender when matching receive is posted**
- **Fully supports the MPI Progress Rule**
  - **Data moves without making MPI library calls**
  - **Portals takes care of progress**





# Reasons for an Eager Protocol

---

- **Portals are based on expected messages**
- **Don't penalize applications that pre-post receives**
- **Optimize for the common case (pre-posting)**
- **Penalty for not pre-posting is not very high**
  - **Network performance (balanced machine)**
  - **Jobs are placed close together in the network so contention should be local**
- **Previous research had shown eager protocols to be a significant performance advantage**
- **MPI implementation is less complex since progress is maintained outside of the MPI library**





# Motivation

---

- **But is eager better?**
  - No real data that pre-posting is the common case
  - Not sure whether the eager optimization improves application performance
  - Unclear whether packetization is an issue
- **Direct comparison with standard rendezvous might answer these questions**





# Standard Rendezvous Implementation

---

- **Short protocol is the same**
- **Long protocol**
  - **Sends a zero-length message and waits for receiver to read data**
  - **Receiver waits for zero-length message and reads data**
- **Can choose eager or standard rendezvous at runtime via environment variable**





# Added Complexity

---

- **Completion of eager protocol messages only involve the particular message being completed**
- **Standard rendezvous**
  - **Must look at all outstanding communication requests since it cannot block waiting for only one to complete**
  - **Does not comply with the MPI Progress Rule**
    - **(Or is less efficient since data can only move when MPI library calls are made)**





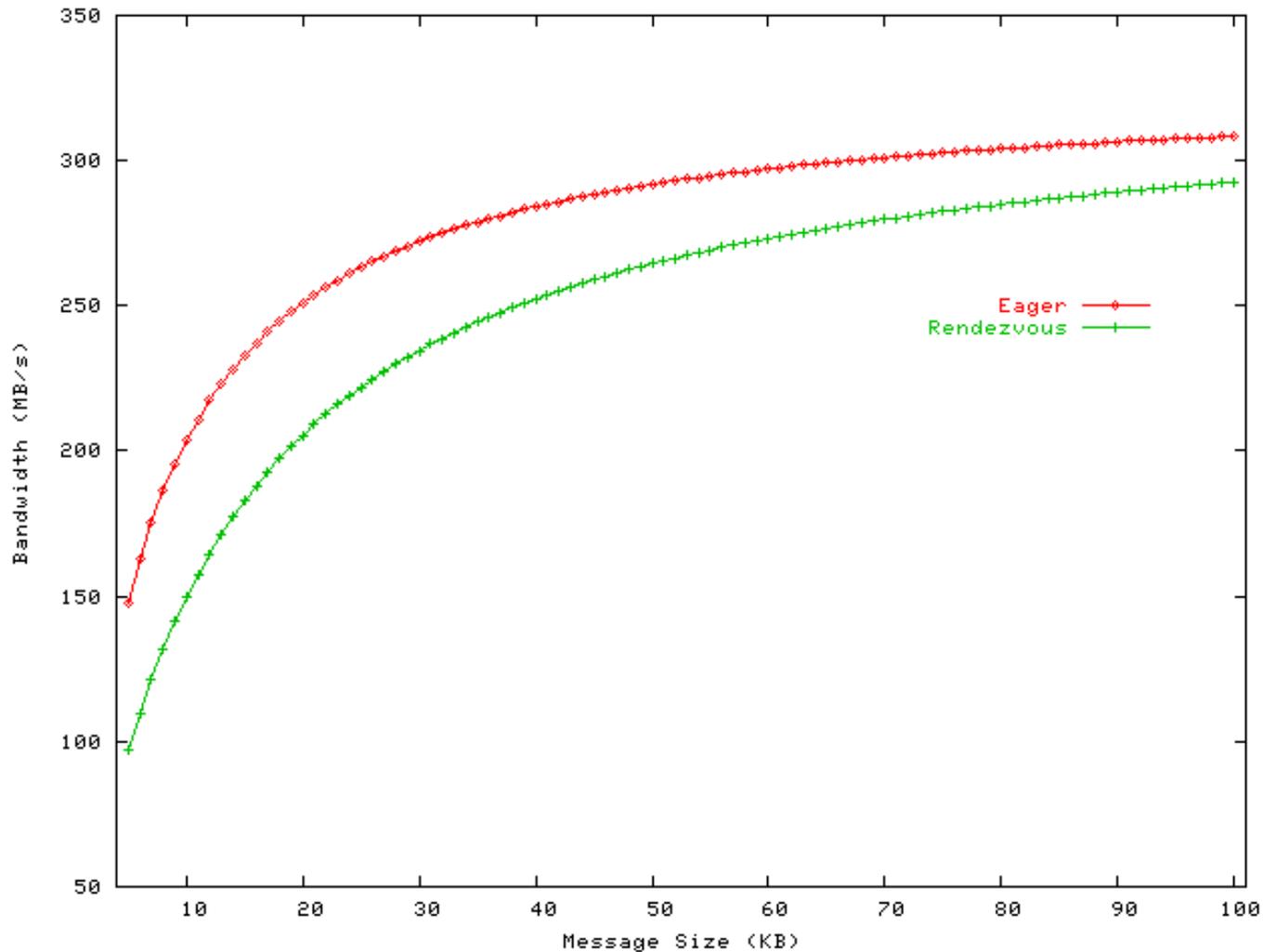
# Micro-Benchmarks

---

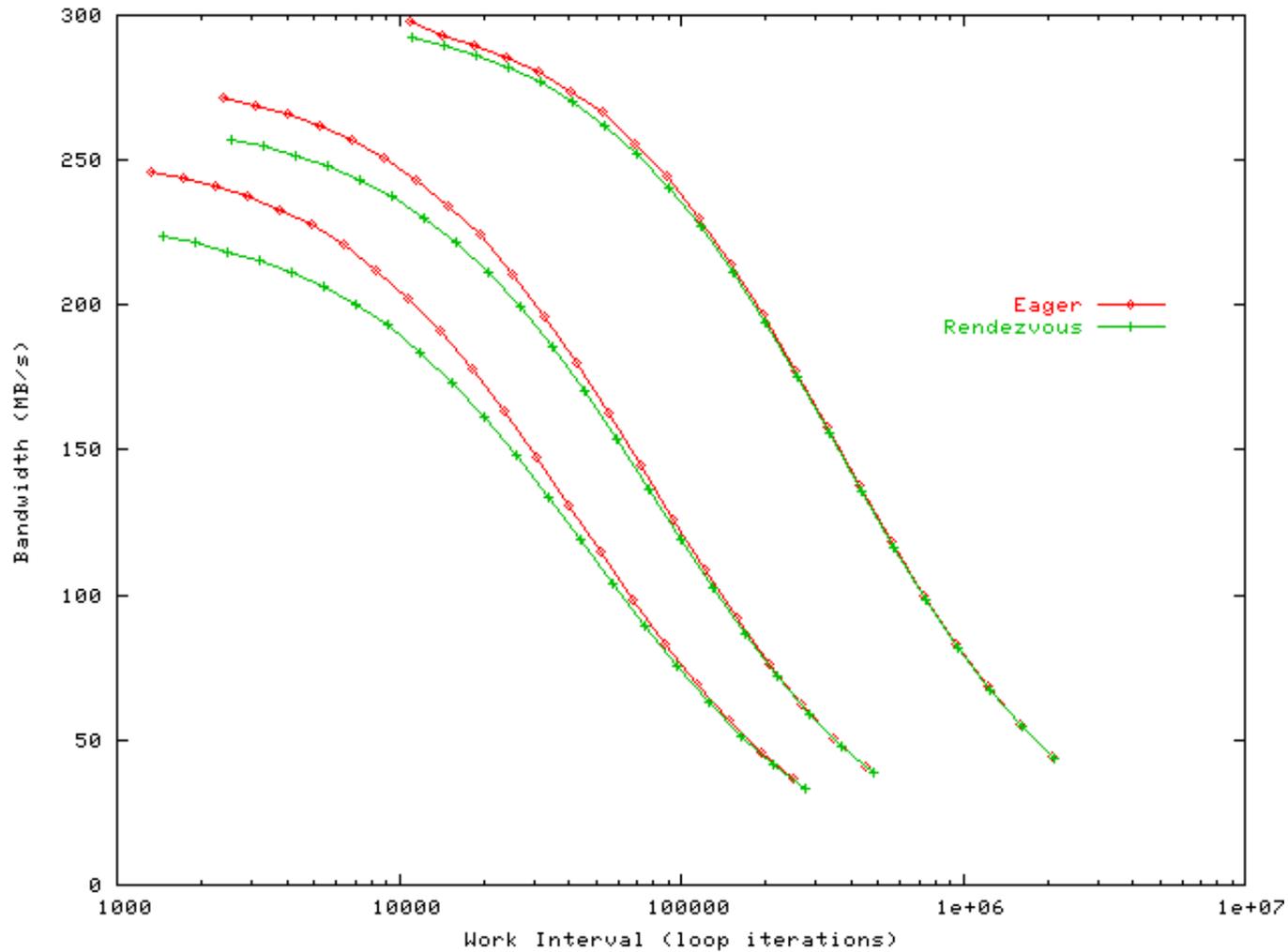
- **Ping-pong bandwidth test**
  - Pre-posted receives
- **Post-Work-Wait (PWW) method of the COMB benchmark suite**
  - Calculates effective bandwidth for a given simulated work interval
  - Used to measure the achievable overlap of computation and communication
  - Pre-posted receives
- **NetPIPE**
  - Determines aggregate throughput by exchanging ping-pong messages for a fixed period of time
  - Pipeline test that sends several messages in a burst (ping(n)-pong)
  - No guarantee of pre-posting



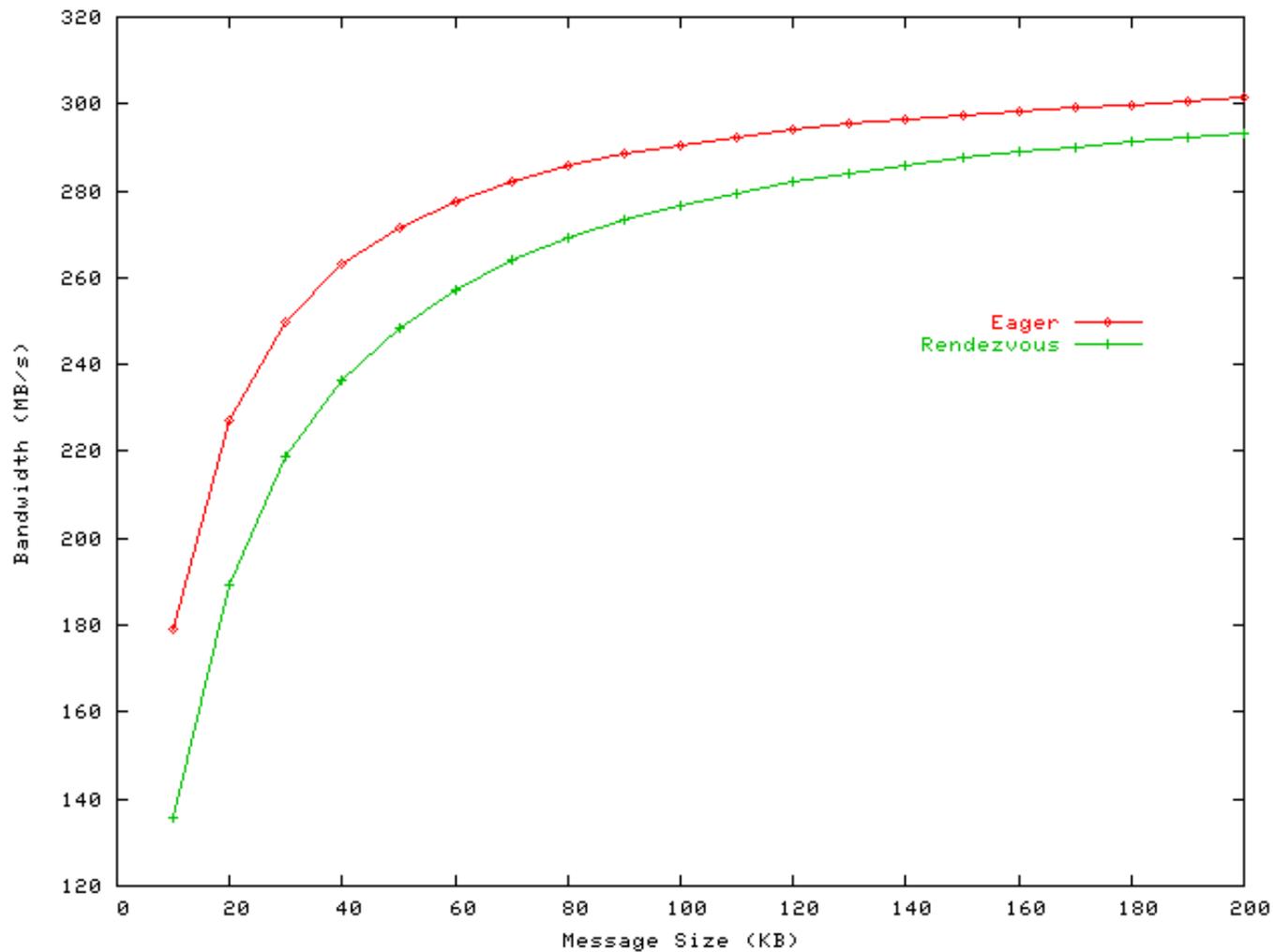
# Ping-Pong Performance



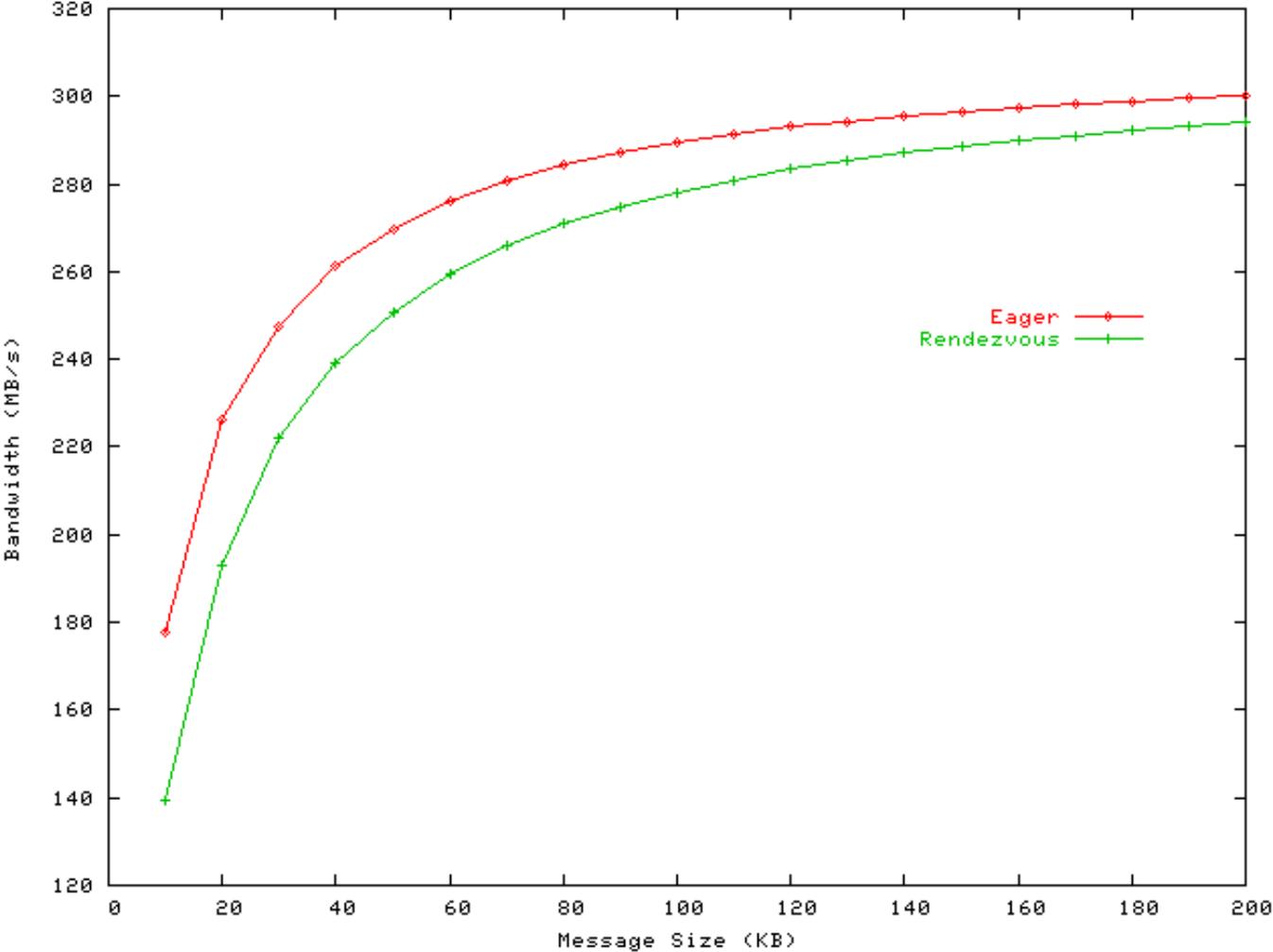
# PWW Performance



# NetPIPE Ping-Pong



# NetPIPE Pipeline





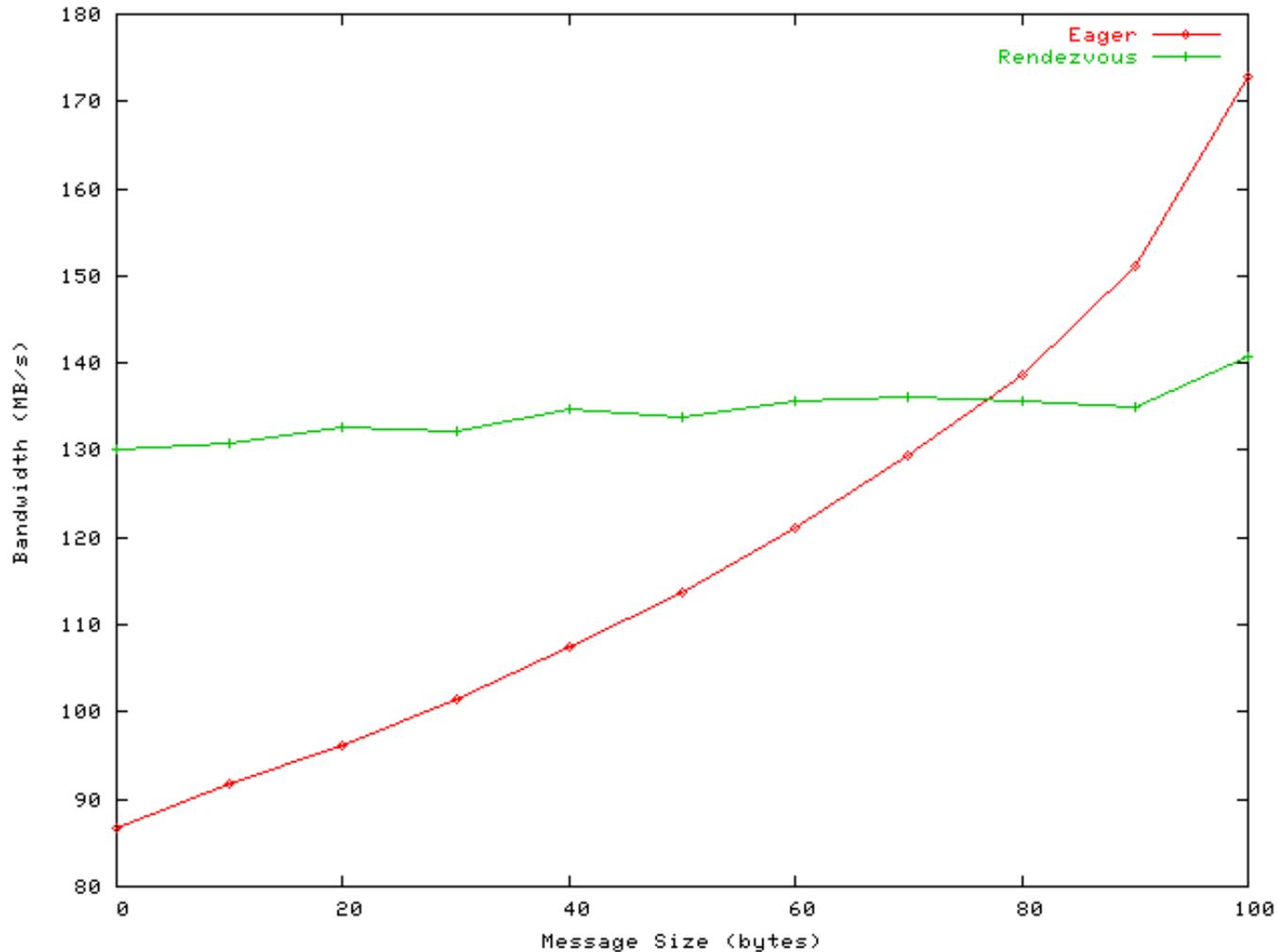
# New Micro-Benchmark

---

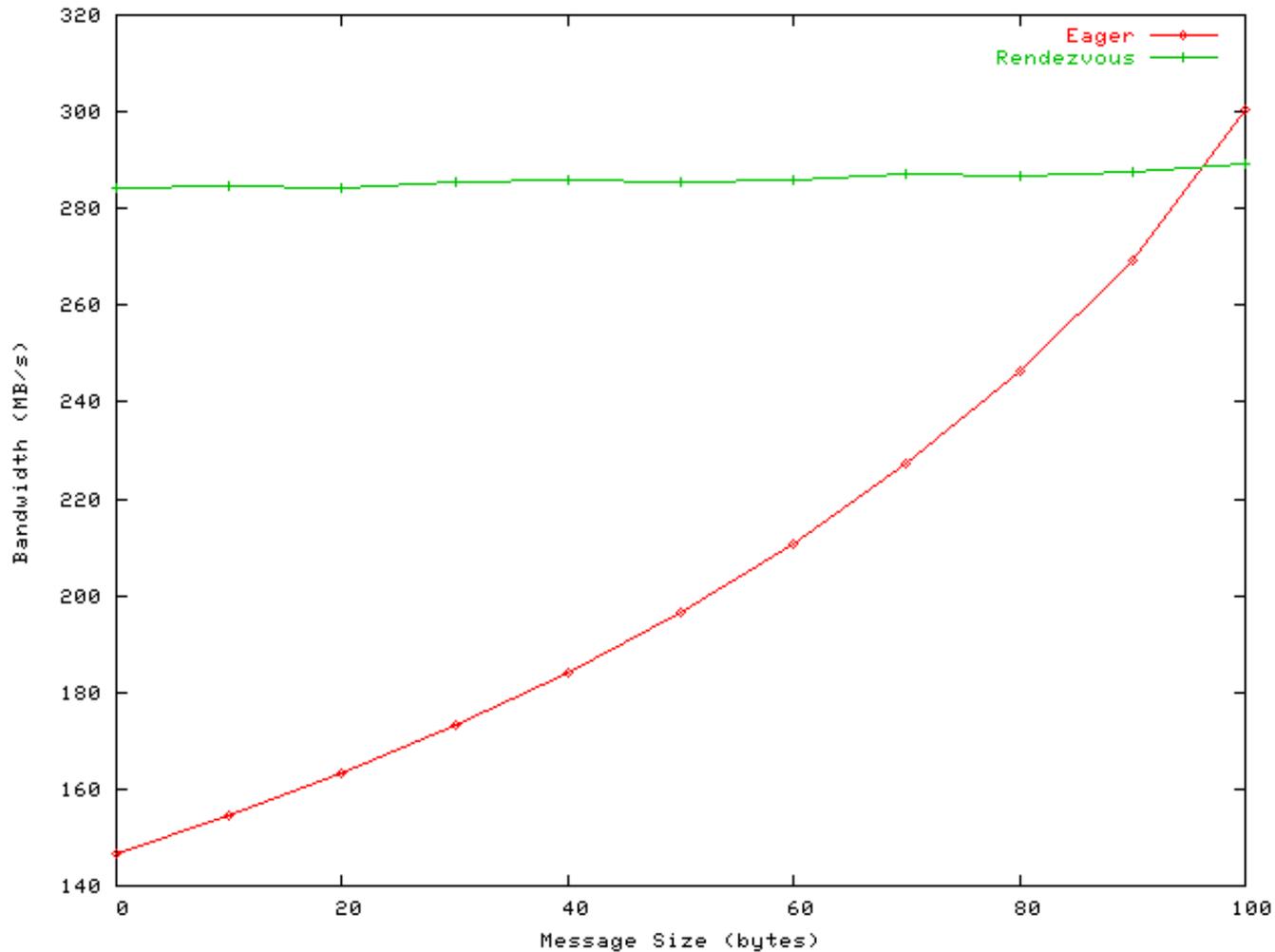
- **Only NetPIPE considers unexpected messages**
- **Real applications are likely to have mixture of pre-posted and unexpected messages**
- **How do unexpected messages impact bandwidth?**
  
- **We designed a new micro-benchmark that measures bandwidth with a parameterized percentage of pre-posted receives**



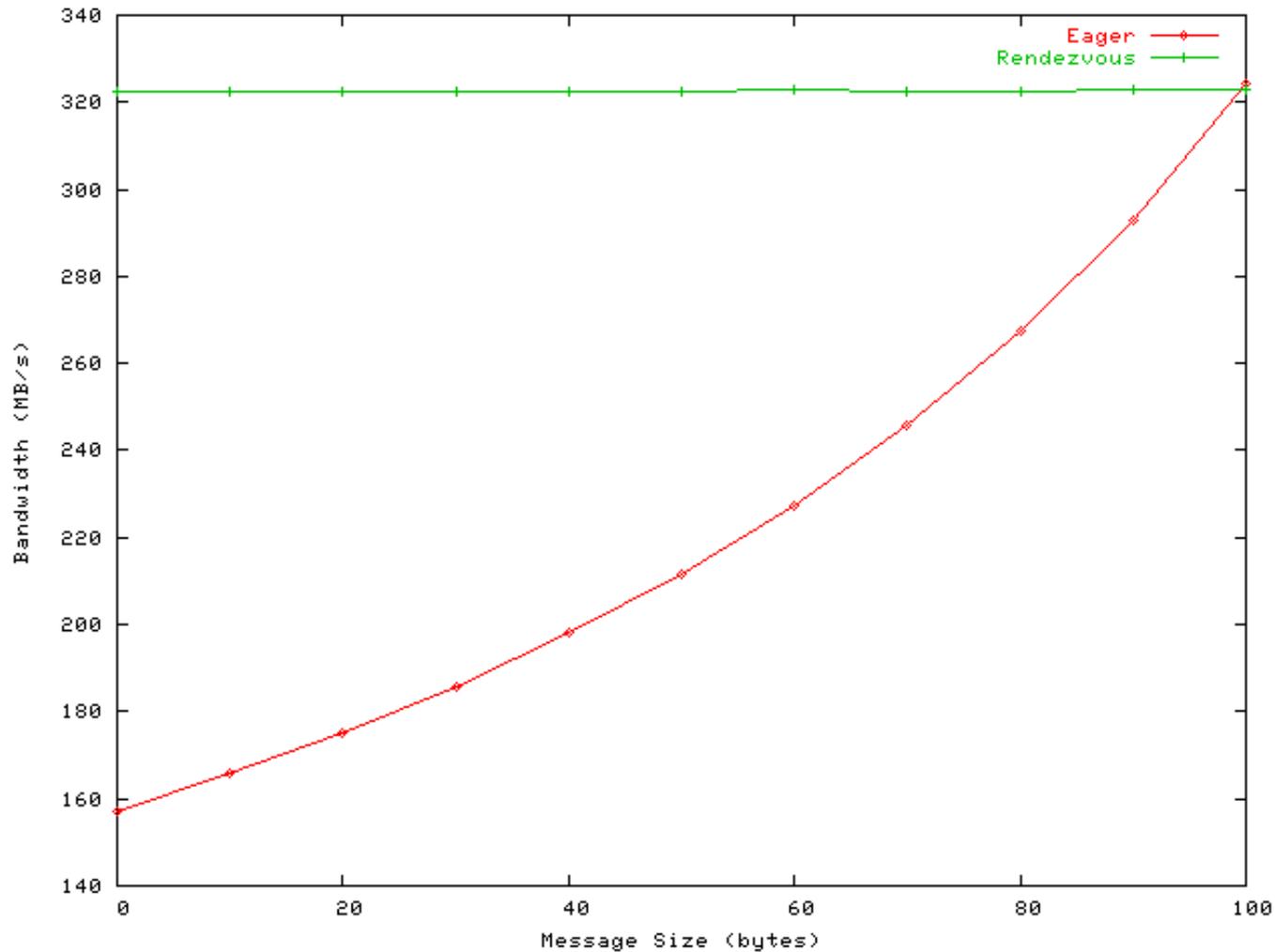
# Performance for 10KB Messages



# Performance for 100KB Messages



# Performance for 1MB Messages





---

**So eager is better, right?**





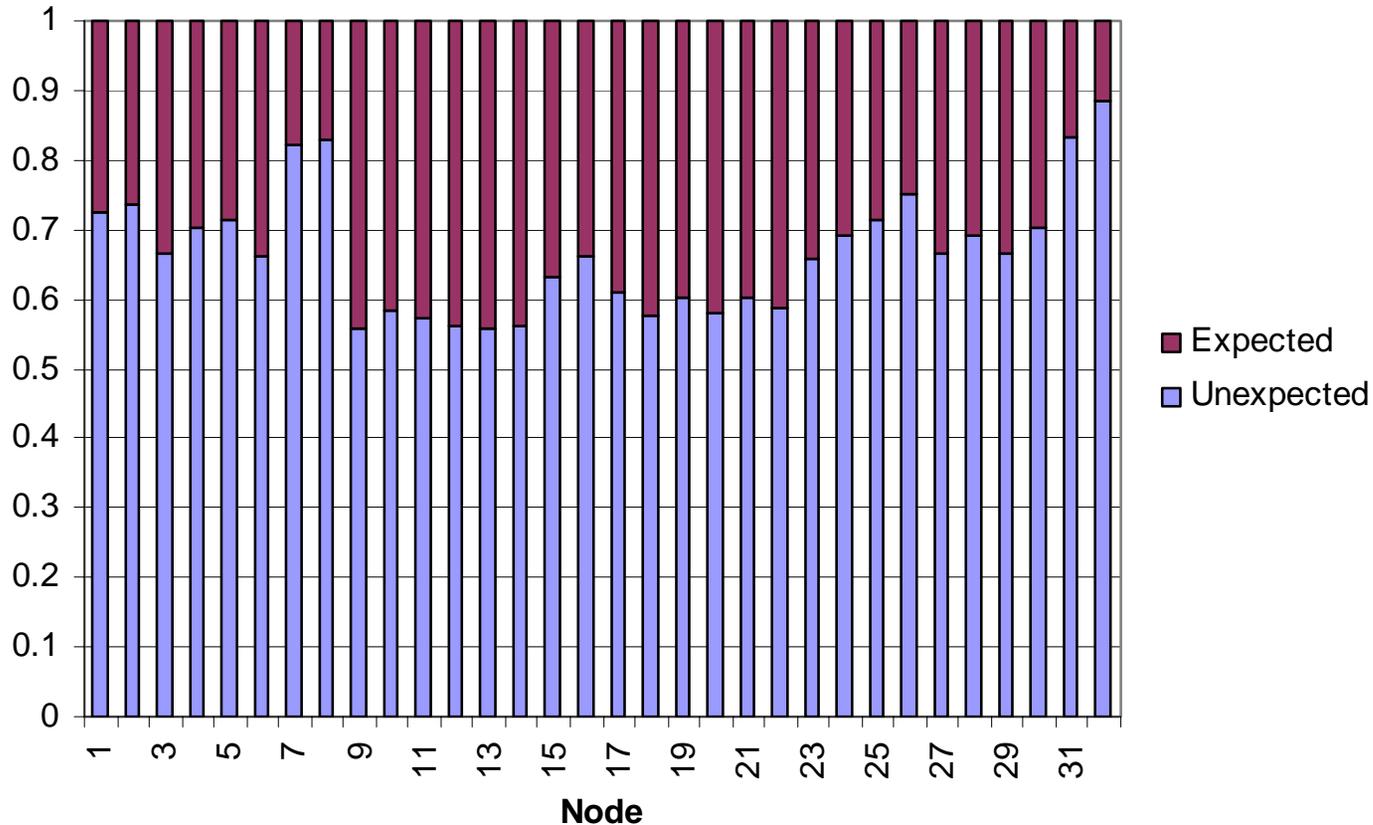
# CTH Application

---

- **Models complex multi-dimensional, multi-material problems characterized by large deformations and/or strong shocks**
- **Uses two-step, second-order accurate finite-difference Eulerian solution**
- **Material models for equations of state, strength, fracture, porosity, and high explosives**
- **Impact, penetration, perforation, shock compression, high explosive initiation and detonation problems**



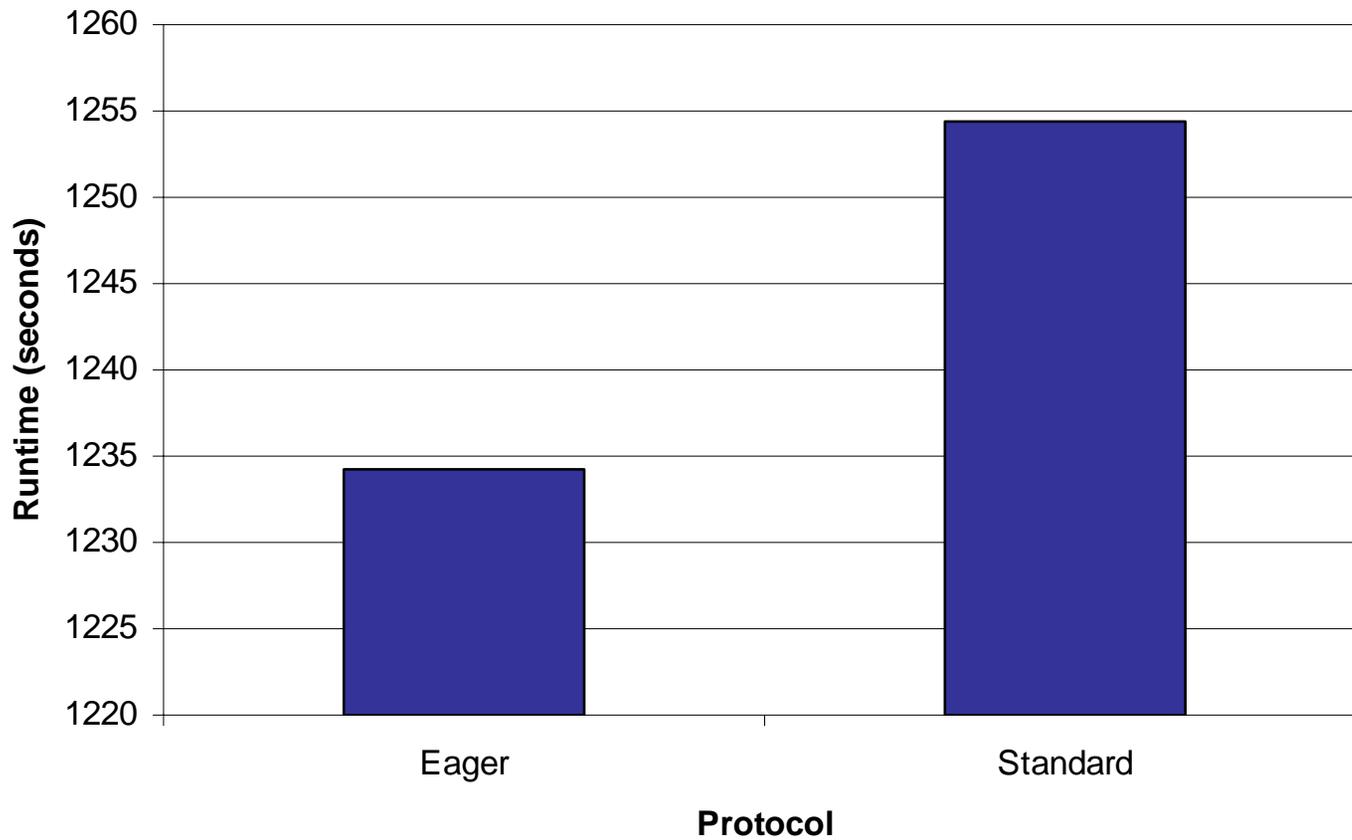
# Expected vs. Unexpected Messages



\*Average  
msg  
size is  
500KB



# CTH Runtime





# CTH Results

---

- **Performance is better with the eager rather than standard rendezvous**
- **New micro-benchmark doesn't consider independent progress, which could be important**
- **Need to do more analysis of real applications to understand implications of unexpected messages and independent progress**





# Summary

---

- **Described an implementation of a standard rendezvous and eager rendezvous**
- **Initial premise was that eager rendezvous had a significant advantage over standard rendezvous**
- **Standard rendezvous is more complex and does not support the MPI Progress Rule (or is less efficient at independently moving data)**
- **Presented a new micro-benchmark and results**
- **Application results show that micro-benchmarks don't tell the whole story**

