Using GPUs to Enable Highly Reliable Embedded Storage

Matthew Curry (curryml@cis.uab.edu)
Lee Ward (lee@sandia.gov)
Anthony Skjellum (tony@cis.uab.edu)
Ron Brightwell (rbbrigh@sandia.gov)

University of Alabama at Birmingham
115A Campbell Hall
1300 University Blvd.
Birmingham, AL 35294-1170

Computer Science Research Institute
Sandia National Laboratory
PO Box 5800
Albuquerque, NM 87123-1319

High Performance Embedded Computing (HPEC) Workshop

23-25 September 2008

The Storage Reliability Problem

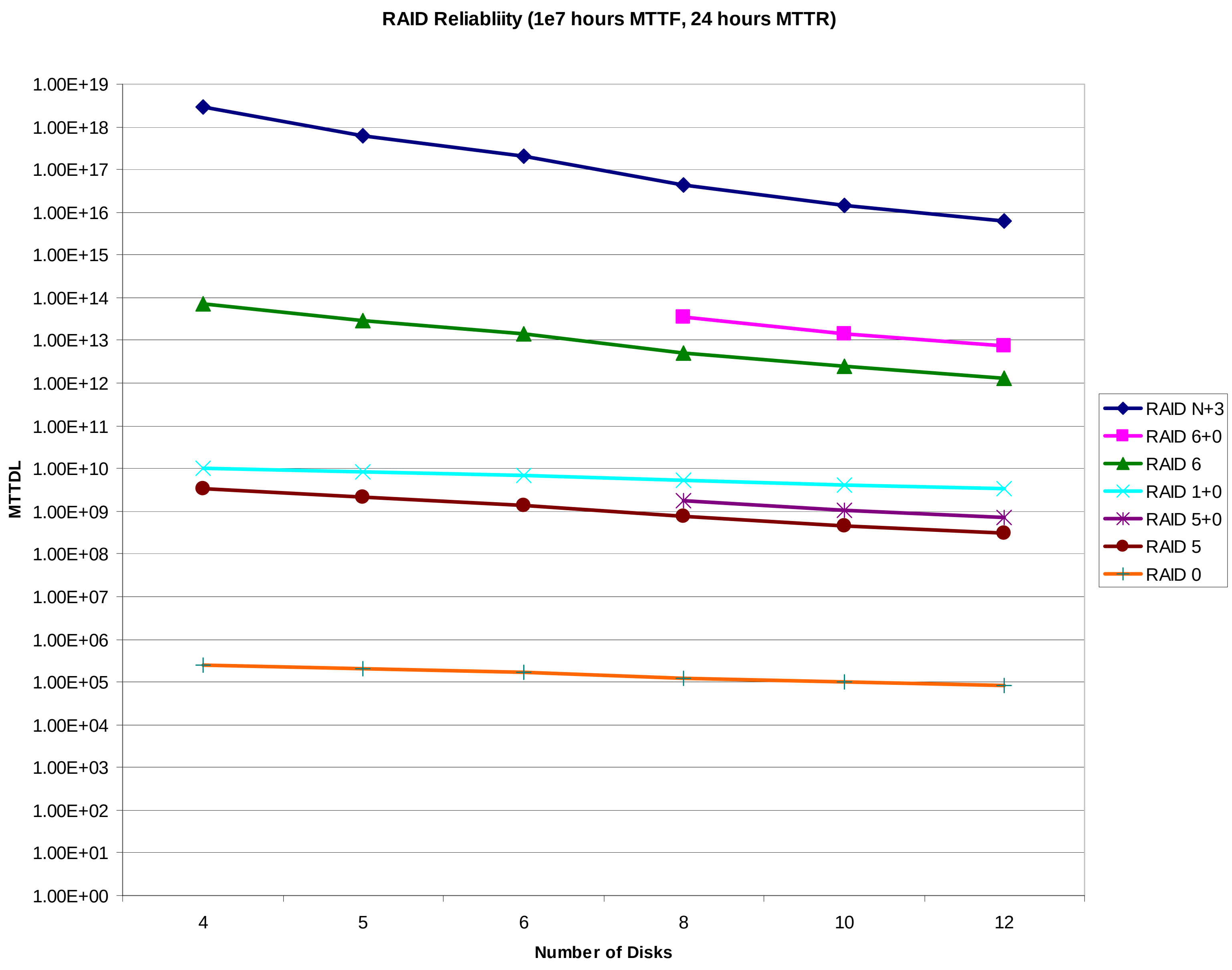
- Embedded environments are subject to harsh conditions where normal failure estimates may not apply
- Since many embedded systems are purposed for data collection, data integrity is of high priority
- Embedded systems often must contain as little hardware as possible (e.g. space applications)

Current Methods of Increasing Reliability

- RAID
  - RAID 1: Mirroring (Two-disk configuration)
  - RAID 5: Single Parity
  - RAID 6: Dual Parity
- Nested RAID
  - RAID 1+0: Stripe over multiple RAID 1 sets
  - RAID 5+0: Stripe over multiple RAID 5 sets
  - RAID 6+0: Stripe over multiple RAID 6 sets

Current Methods of Increasing Reliability

- RAID MTTDL (Mean Time to Data Loss)
  - RAID 1: $MTTF^2/2$
  - RAID 5: $MTTF^2/(D*(D-1))$
  - RAID 6: $MTTF^3/(D*(D-1)*(D-2))$
- Nested RAID MTTDL
  - RAID 1+0: MTTDL(RAID1)/N
  - RAID 5+0: MTTDL(RAID5)/N
  - RAID 6+0: MTTDL(RAID6)/N


RAID Reliabliity (1e7 hours MTTF, 24 hours MTTR)
1.00E+19
1.00E+18
1.00E+17
1.00E+16
1.00E+15
1.00E+14
1.00E+13
1.00E+12
1.00E+11
1.00E+10
1.00E+09
1.00E+08
1.00E+07
1.00E+06
1.00E+05
1.00E+04
1.00E+03
1.00E+02
1.00E+01
1.00E+00
MTTDL
4
5
6
8
10
12
Number of Disks
RAID N+3
RAID 6+0
RAID 6
RAID 1+0
RAID 5+0
RAID 5
RAID 0

Why N+3 (Or Higher) Isn't Done

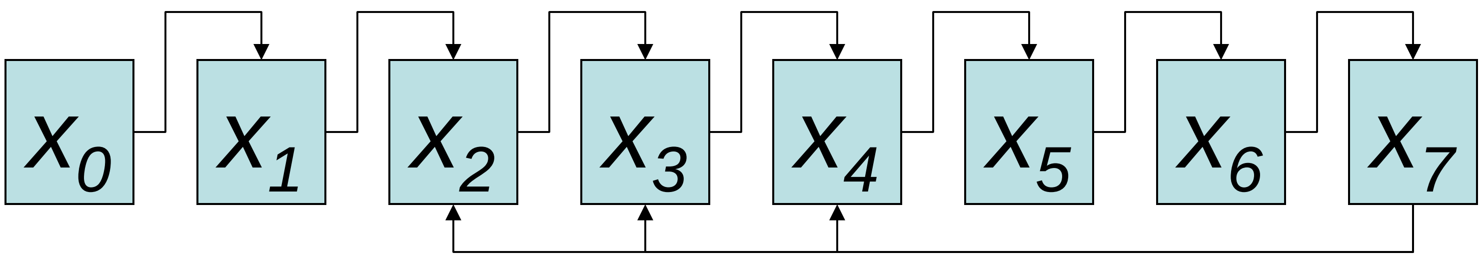
- Hardware RAID solutions largely don't support it
  - Known Exception: RAID-TP from Accusys uses three parity disks
- Software RAID doesn't support it
  - Reed-Solomon coding is CPU intensive and inefficient with CPU memory organization

An Overview of Reed-Solomon Coding

- General method of generating arbitrary amounts of parity data for $n+m$ systems
- A vector of n data elements is multiplied by an n x m dispersal matrix, yielding m parity elements
- Finite field arithmetic

Multiplication Example

- $\{37\} = 32 + 4 + 1 = 100101 = x_5 + x_2 + x_0$
- Use Linear Shift Feedback Register to multiply an element by {02}



Multiplication Example

- Direct arbitrary multiplication requires distributing so that only addition (XOR) and multiplication by two occur.
  - {57} x {37}
  - {57} x ({02}^{5} + {02}^{2} + {02})
  - {57} x {02}^{5} + {57} x {02}^{2} + {57} x {02}
- Potentially dozens of elementary operations!

Optimization: Lookup Tables

- Similar to the relationship that holds for real numbers:

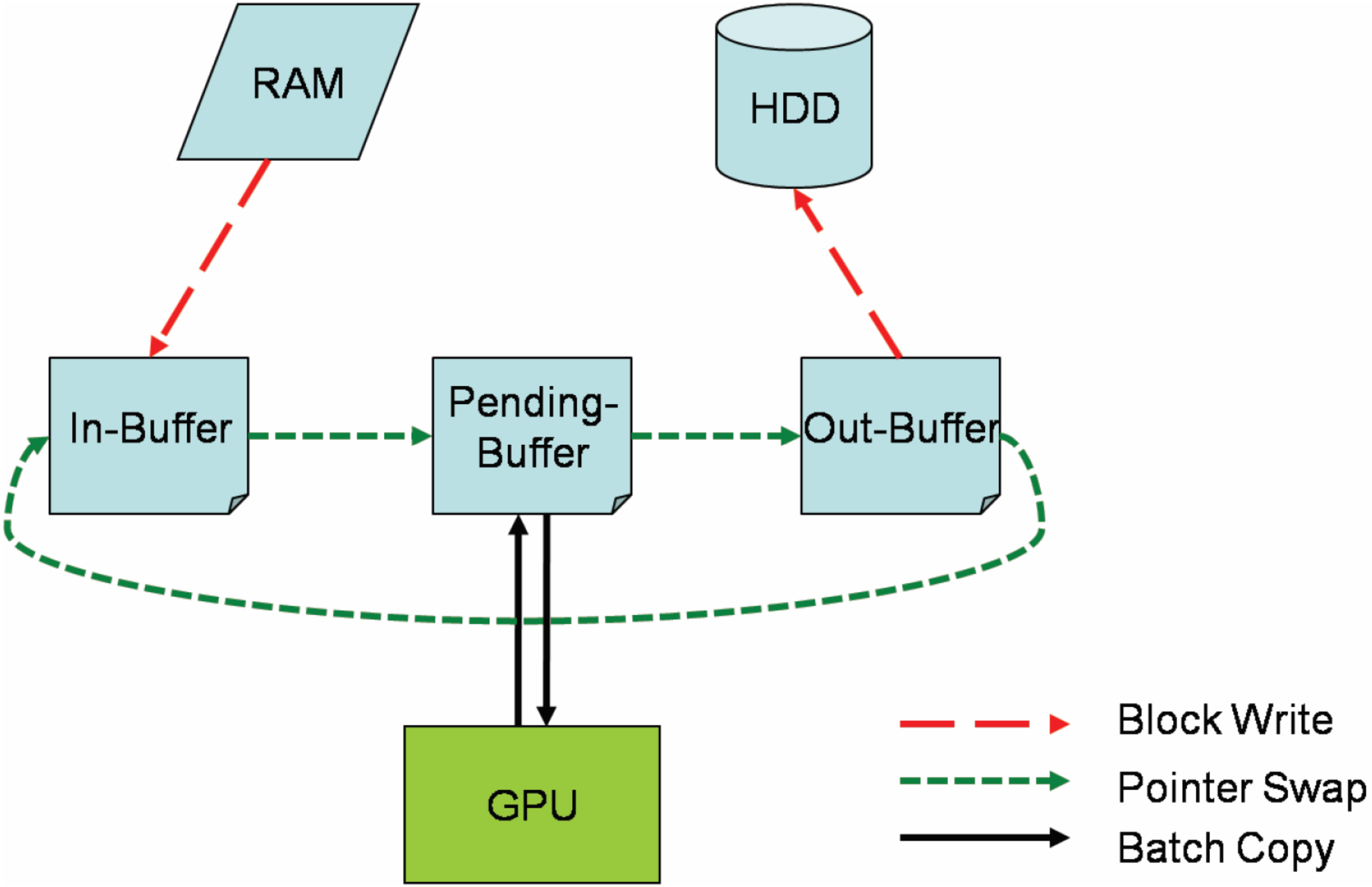
$$e^{log(x)+log(y)} = x * y$$

- This relationship translates (almost) directly to finite field arithmetic, with lookup tables for the logarithm and exponentiation operators
- Unfortunately, parallel table lookup capabilities aren't common in commodity processors
  - Waiting patiently for SSE5

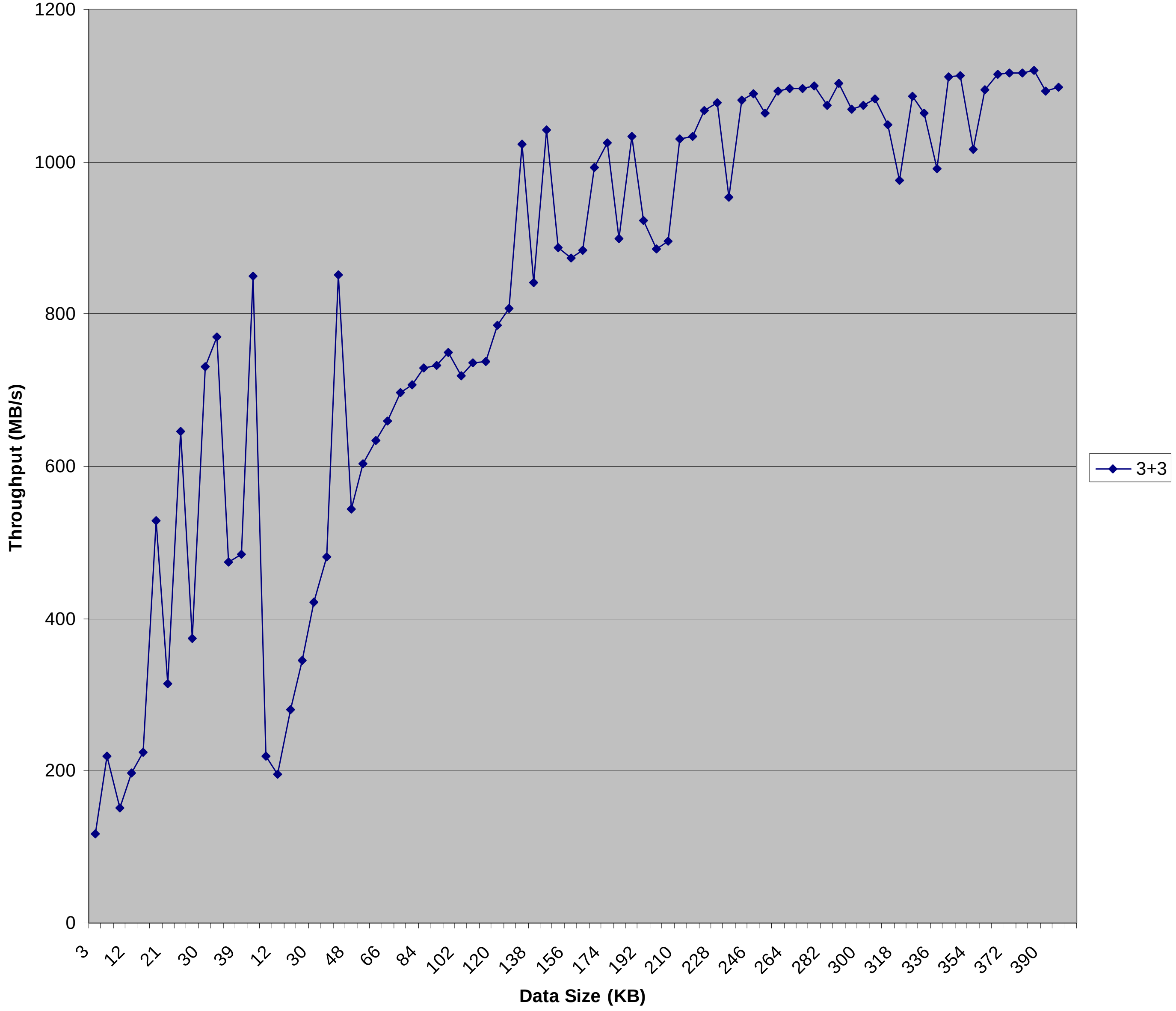
NVIDIA GPU Architecture

- GDDR3 Global Memory
- 16-30 Multiprocessing Units
- One shared 8 KB memory region per multiprocessing unit (16 banks)
- Eight cores per multiprocessor

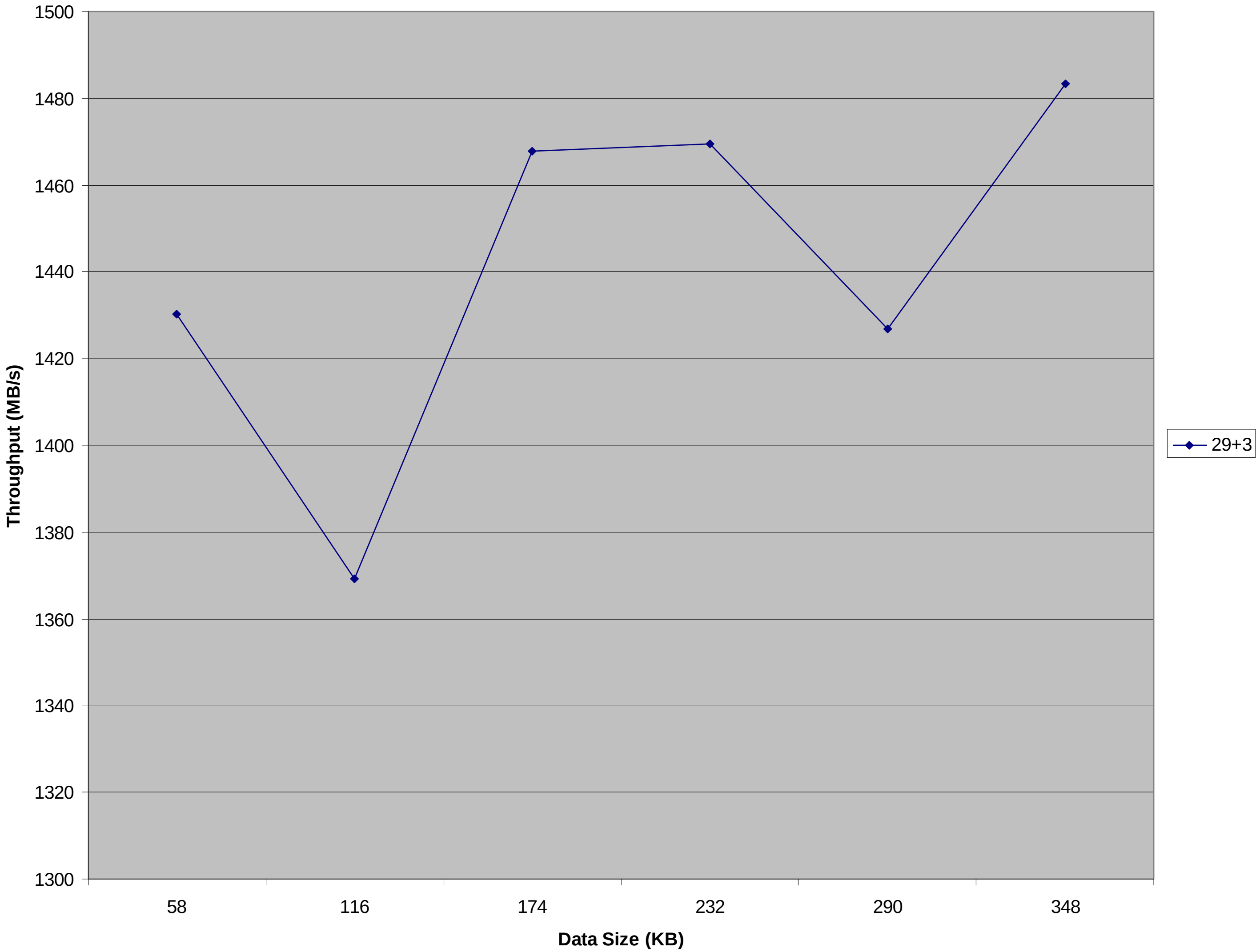
Integrating the GPU



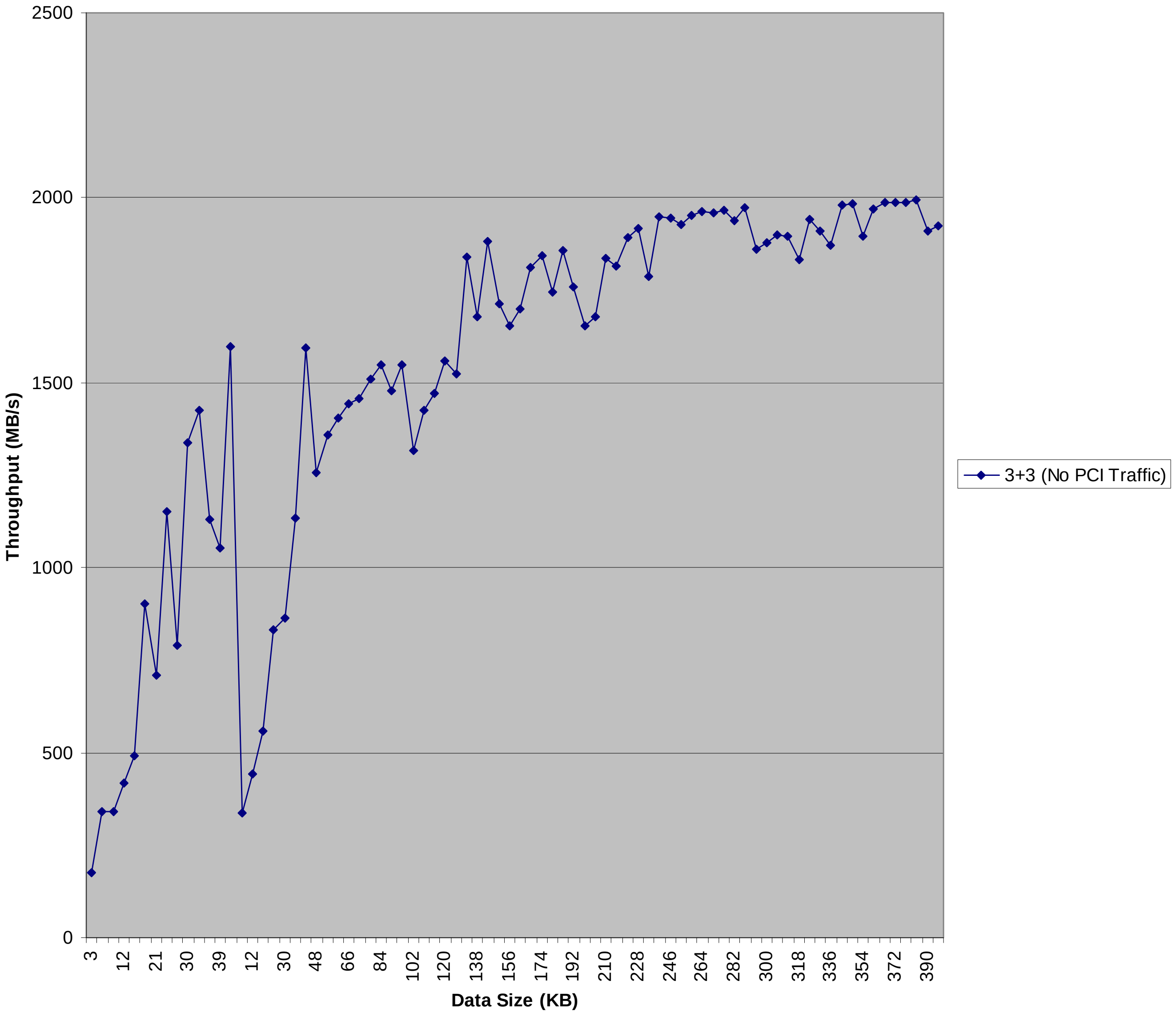
3+3 Performance



29+3 Performance



Neglecting PCI Traffic: 3+3



Conclusion

- GPUs are an inexpensive way to increase the speed and reliability of software RAID
- By pipelining requests through the GPU, N+3 (and greater) are within reach
  - Requires minimal hardware investment
  - Provides greater reliability than available with current hardware solutions
  - Sustains high throughput compared to modern hard disks