Analyzing the Scalability of Eldorado

Keith D. Underwood
Megan Vance

presented at CUG 2006 by
Jon Berry
(Discrete Math and Algorithms Dept.)

May 11, 2006

Scalable Computing Systems Dept
Sandia National Laboratories

Motivation

- **Graph algorithms perform extremely well on the MTA-2**
  - Won IC graph benchmarking contest in 2004
  - Latency tolerance is key
- **The MTA-2 can be "easier" to program for graph algorithms than using MPI on a distributed memory machine**
  - Generic software infrastructures hide some of shared-memory complexity
  - Much easier to handle many classes of input in a generic way
- **Graph algorithms are of interest**
  - Pattern finding in relational data could become a kernel of counter-terrorism work

Massive Multithreading: The Cray MTA-2

- **Slow clock rate (220Mhz)**
- **128 "streams" per processor**
- **Global address space**
- **Fine-grain synchronization**
- **Simple, serial-like programming model**
- **Advanced parallelizing compilers**

No Processor Cache

Latency Tolerant:

important for Graph Algorithms

Hashed Memory

Introduction to Eldorado

- **The MTA-2 has amazing performance on graph algorithms, but doesn't scale to large enough sizes**

- **Building a scalable infrastructure is expensive**
  - Board design, cabinet design, signal integrity work
  - Scalable management software infrastructure

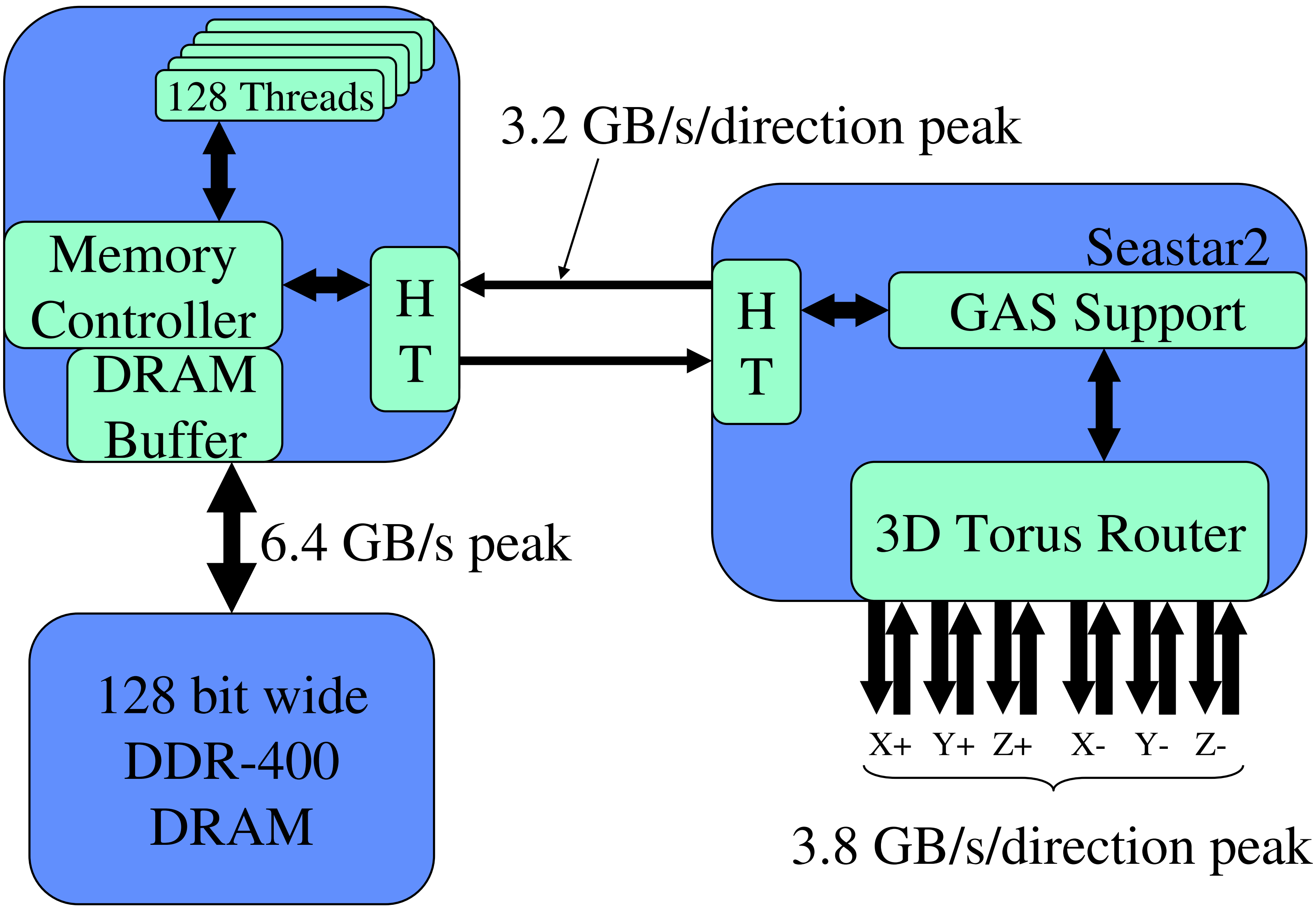
- **Low cost approach to an MTA-2 successor: leverage the XT3**
  - Refresh the MTA-2 design to run at 500MHz
  - Put it in an Opteron socket

Primary Changes: Eldorado vs. MTA-2

	MTA-2	Eldorado (512 Node)
Clock Rate	220 MHz	500 MHz
Topology	Modified-Cayley	3D-Torus
Memory	Full speed random access	Standard DRAM
Memory Rate (Best)	220 MW/s	500 MW/s
Memory Rate (Worst, Remote)	220 MW/s	75 MW/s
Memory Rate (Worst, Local)	None	100 MW/s
Data “Cache” (DRAM buffer)	None	128 KB, 64 byte lines
Bisection BW	3.5GB/s * P	15.3 GB/s * $P^{2/3}$

Sandia National Laboratories

New System Picture



Major Question: Will Eldorado still Scale?

- **Every change to the hardware could have a negative impact on performance for graph algorithms**
  - New memory system has poor random access characteristics
  - The network is not designed for an MTA-2 processor (reduced relative bisection bandwidth)
  - Graph codes are not traditionally cache friendly – may not be DRAM buffer friendly

- **Potential new bottlenecks**
  - If the “buffer” does not work, a DRAM will not feed a processor
  - The per processor network bisection bandwidth shrinks with scale
  - The relative network latency is much higher and will go up under load

Start with Graph Kernels

- **Connected components**
  - Kahan's three phase algorithm
  - The "bully" algorithm
- **Subgraph Isomorphism**
  - Compound type filtering
  - SNL walk heuristic
- **S-T Connectivity**
  - Bi-directional BFS

Measure and then Simulate

- **We do not have a full system simulator, but we can simulate the pieces**
- **Measure the graph kernels**
  - How often do they access memory?
  - How much of that is local/remote?
- **Simulate the DRAM buffer**
  - Will the DRAM buffer hit rate be sufficient?
  - What are the impacts of network traffic pollution?
- **Simulate the network**
  - How will the network respond under load?
  - Where will the limitations arise?

Application Measurements

MTGL Kernel	% Memory References	% Stack	Access Rate (Mref/s)		
			Total	Global	Local
Connected Comp: Bully	59	46	295	159	136
Connected Comp: Kahan	60	53	300	141	159
Connected Comp: Simple	56	49	230	117	113
S-T Connectivity: Small	75	10	375	338	37
S-T Connectivity: Medium	60	28	300	216	84
S-T Connectivity: Large	60	32	300	204	96
Sparse Matrix Vector	46	53	230	108	122
Subgraph Isomorphism	30	34	150	99	51

DRAM Buffer Simulation Approach

- **Gather traces from MTA-2 "Zebra" simulator**
  - Cray collected traces from each kernel
  - Traces represent "1 processor" of data
- **Replicate traces as needed**
  - Single processor trace may not be representative of real work load
  - More threads may be needed in Eldorado
  - Traces were assumed to be representative of "some threads"
- **Create as realistic of an environment as possible**
  - Polluting traffic from the network
  - Interleaved requests to the network that were constrained by lookahead

DRAM Buffer Simulation Results: DRAM Buffer Hit Rate

MTGL Kernel	Replications			
	64X	16X	4X	1X
Connected Comp: Bully	20%	63%	85%	99%
Connected Comp: Kahan	13%	52%	79%	92%
Connected Comp: Simple	17%	56%	82%	92%
S-T Connectivity	85%	95%	99%	99%
Sparse Matrix Vector	70%	85%	93%	99.9%
Subgraph Isomorphism	63%	69%	85%	87%

Why the DRAM Buffer Matters

- **The network is going to be a bottleneck**
  - It may deliver 75 Mref/s, but will still be the constraint
  - If it delivers 75 Mref/s, it will take 75 Mref/s from the DRAM
  - The DRAM only delivers 100 Mref/s

- **Only 25 Mref/s are left for the node**
  - That is, if you don't want the DRAM to be a constraint
  - All of the codes require more than this
  - Anything more it steals from the network

Implications of DRAM Buffer Results: Mref/s Needed from DRAM

MTGL Kernel	Replications			
	64X	16X	4X	1X
Connected Comp: Bully	108	50	20	1.3
Connected Comp: Kahan	138	76	33	12.7
Connected Comp: Simple	94	50	20	9
S-T Connectivity	14	5	1	1
Sparse Matrix Vector	37	18	9	0.1
Subgraph Isomorphism	19	16	8	7

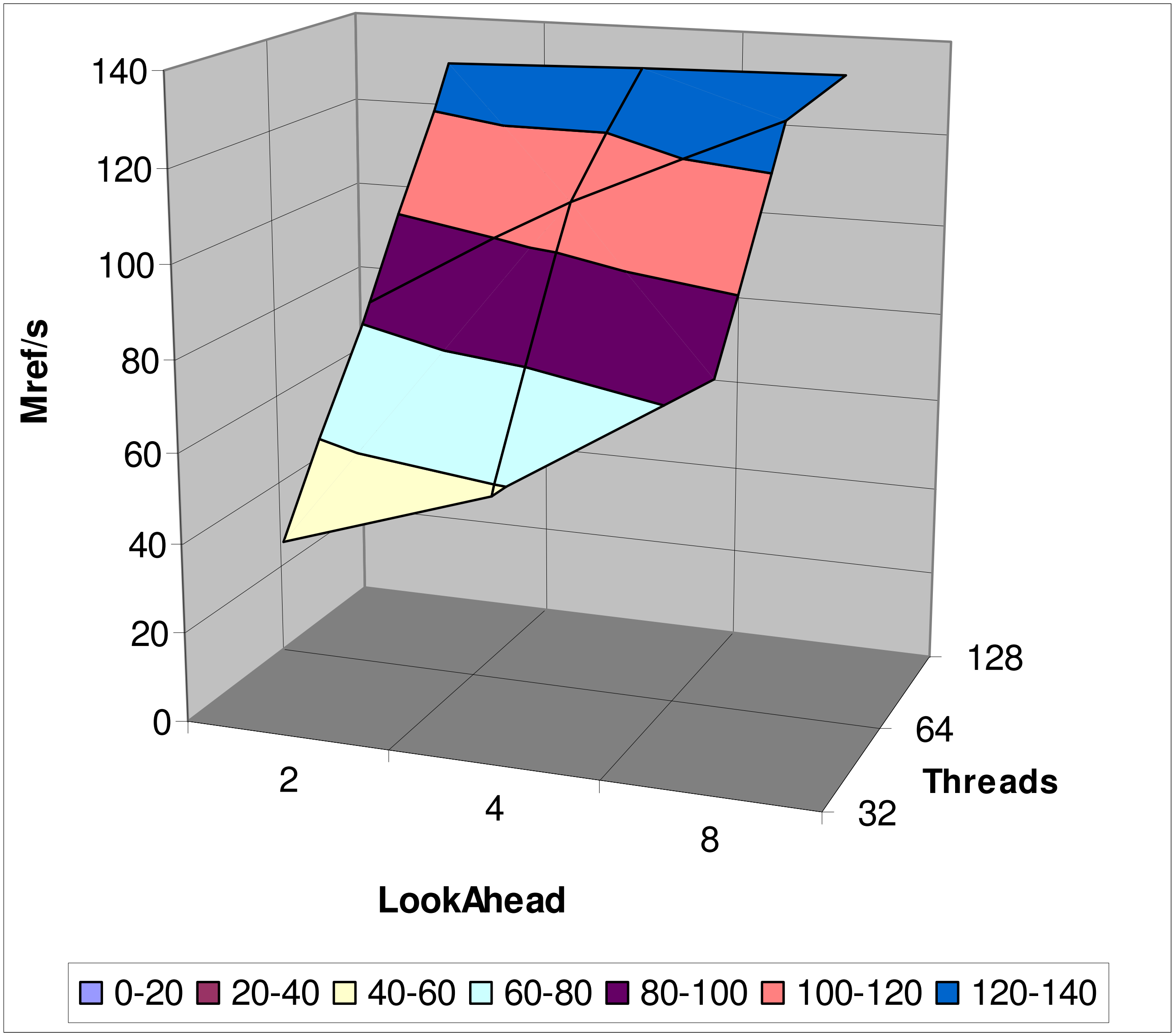
Network Simulation Approach

- **Build a hybrid (cycle based/discrete event) simulation model of the router**
  - Capture as many parameters as possible while maintaining a rational execution time
  - Capture cycle level details of arbitration
- **Drive the network with a statistical model of an Eldorado processor**
  - Subject the statistical model to Eldorado constraints
  - Sweep over parameters of relevance: access rate, local percentage, lookahead, number of threads, DRAM buffer hit rate
  - Currently over 1500 points in that space

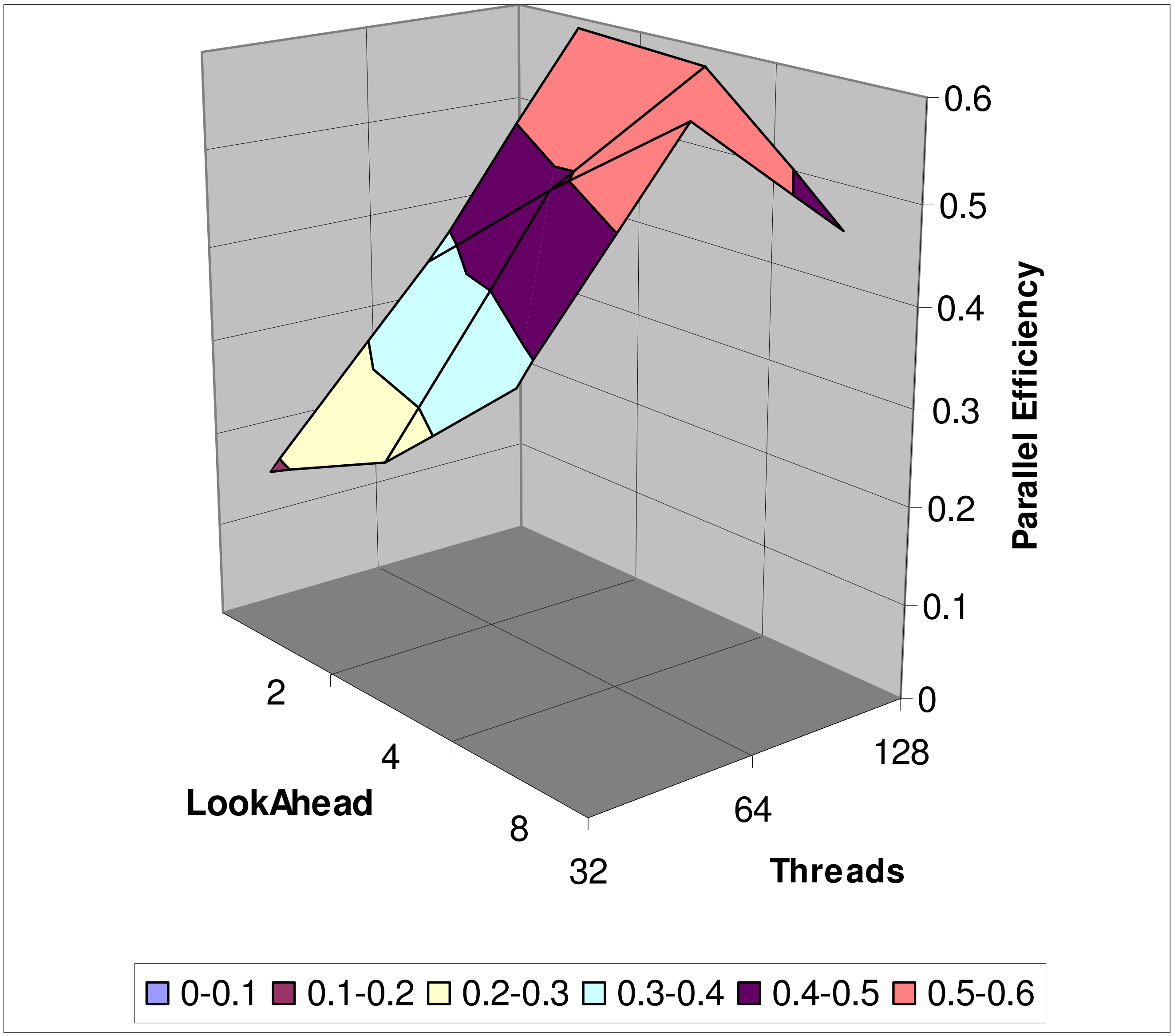
Parameter Definitions

- **Access Rate**: percentage of instructions that access memory times the processor clock rate (500 MHz)
- **Local Percentage**: percentage of instructions that access local DRAM rather than the network
- **LookAhead**: number of instructions between issuing a load and needing the result of the load
- **Number of threads**: simultaneous number of threads assumed per processor
- **DRAM buffer hit rate**: assumed success of the DRAM buffer

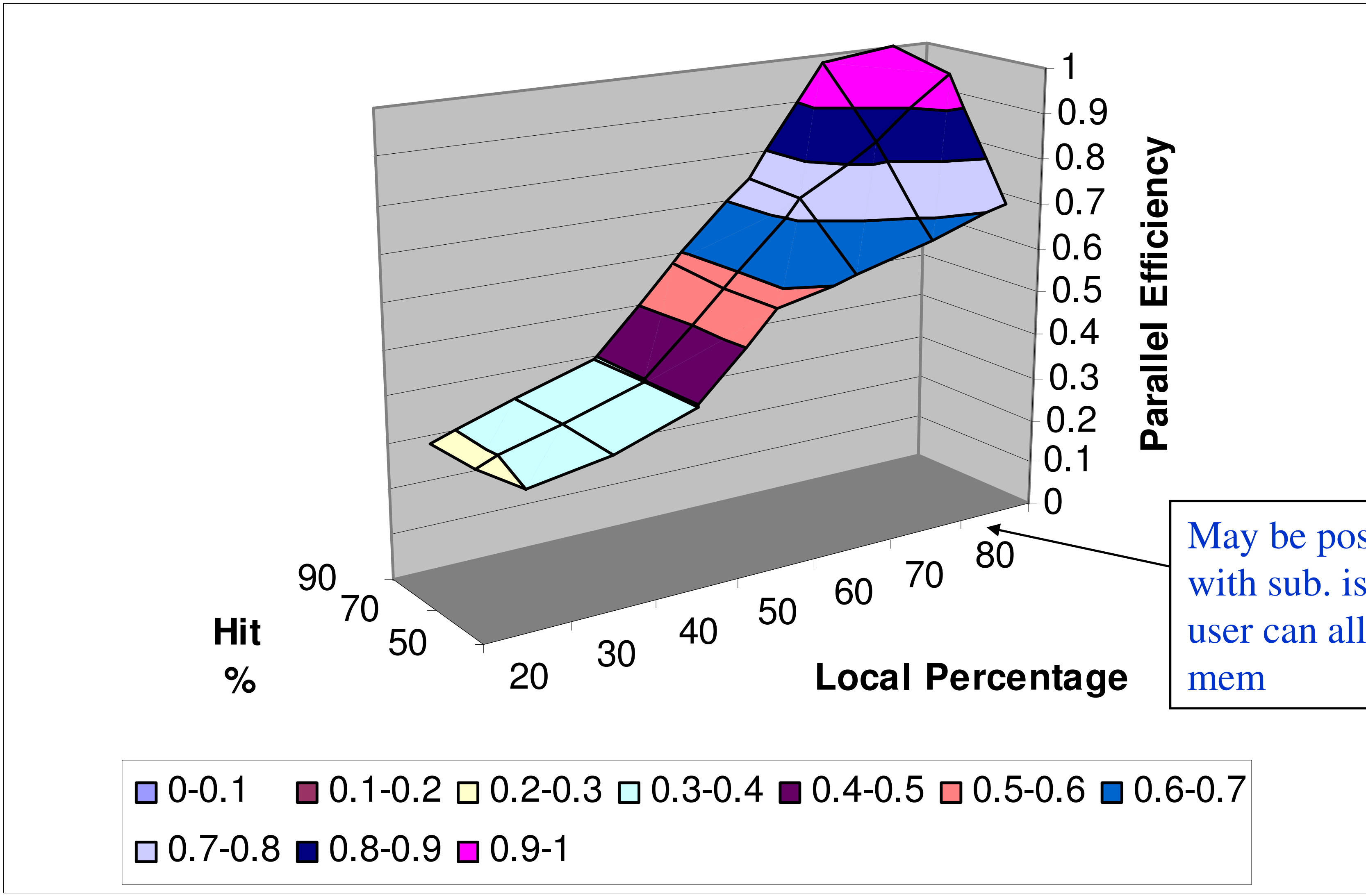
LookAhead and Thread Impacts: Sustained Memory Reference Rate



LookAhead and Thread Impacts: Parallel Efficiency

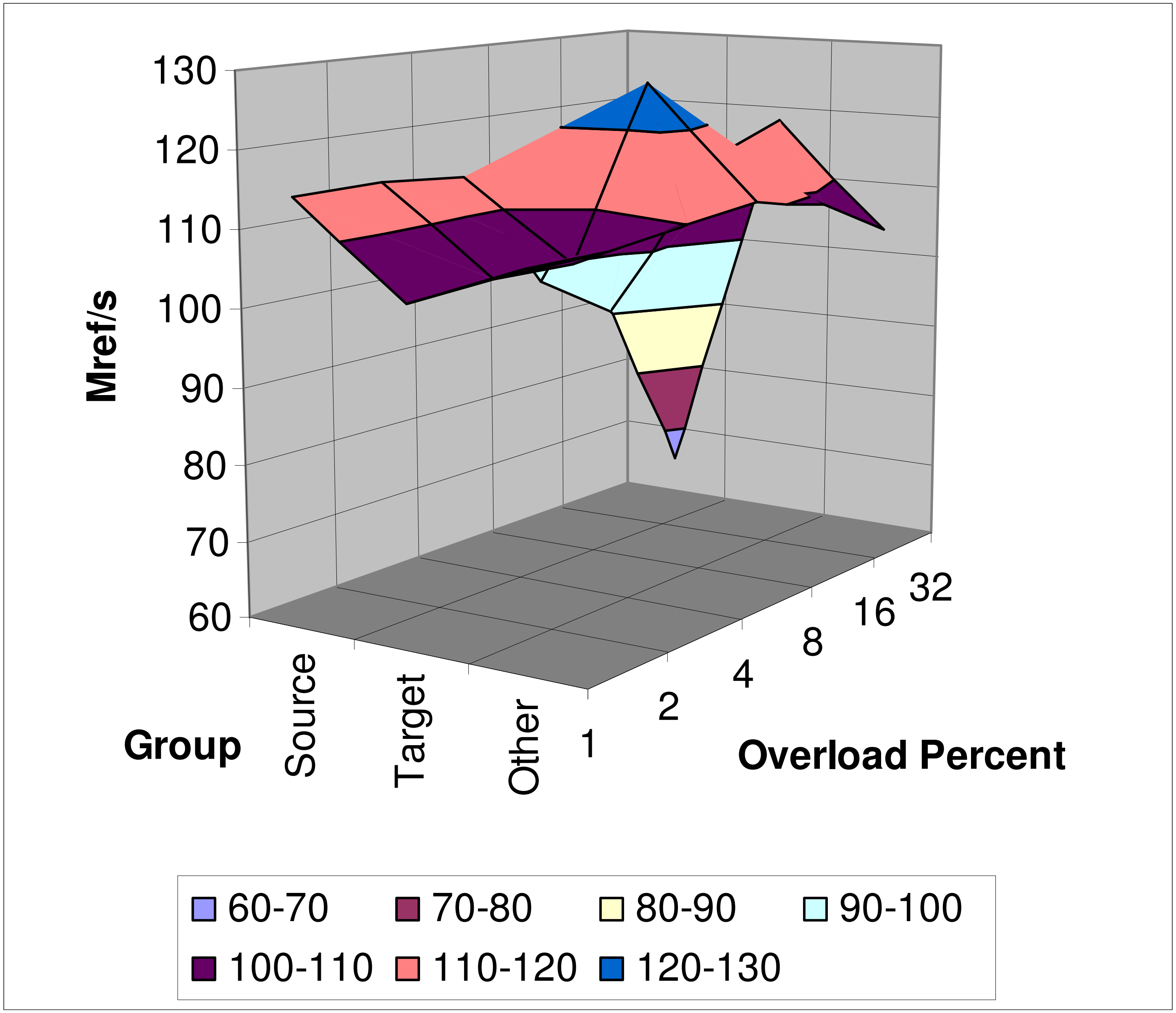


Local Percentage and Hit Rate Impacts: Parallel Efficiency



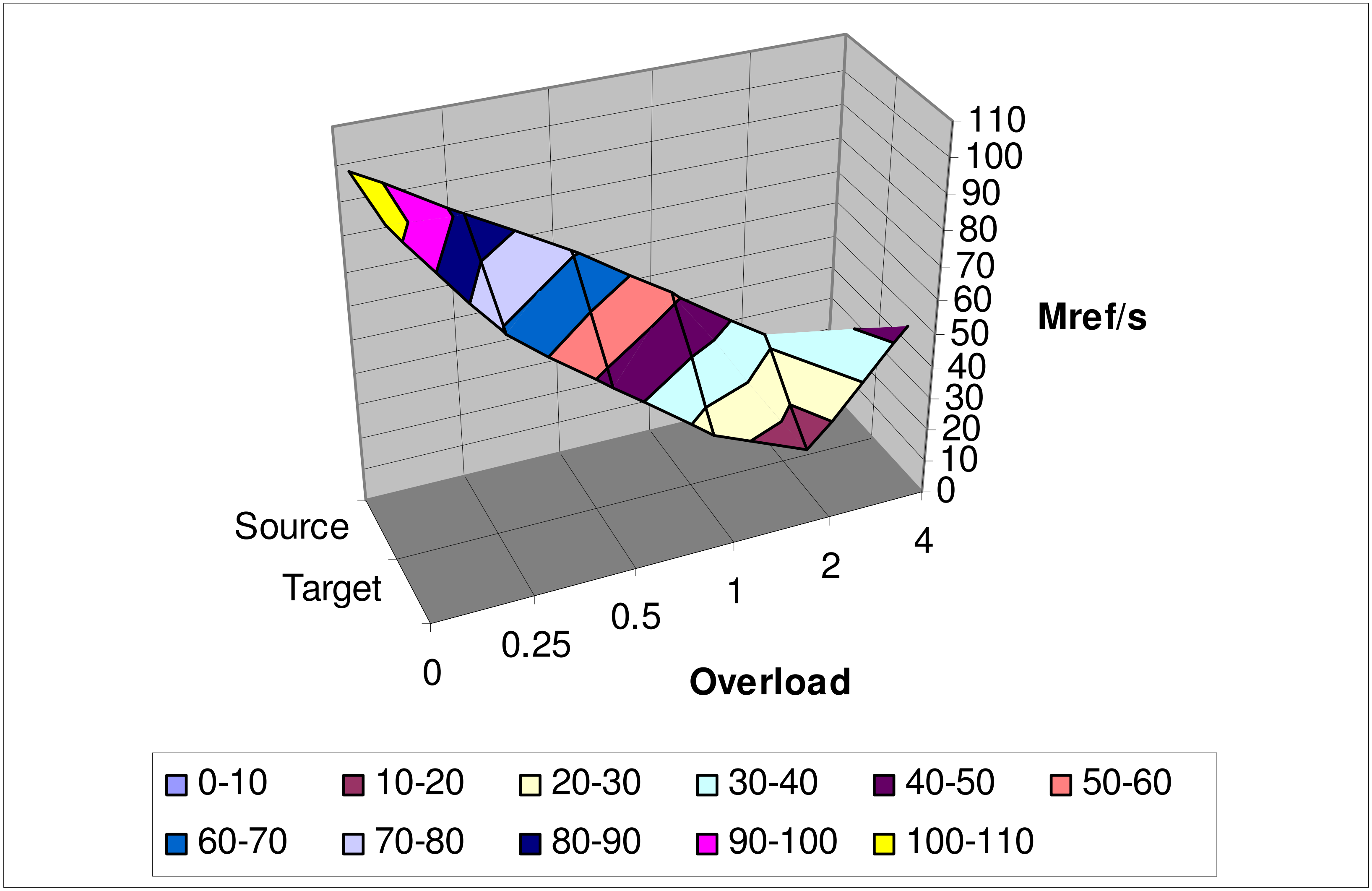
May be possible with sub. iso. if user can alloc local mem

Hot Spot Memory Rate Impacts: Few to Few



Sandia National Laboratories

Hot Spot Memory Rate Impacts: All to Few



Implications for Graph Kernels

- **S-T Connectivity**
  - Currently looks to be the worst scaling of the bunch
  - Particularly bad if not visiting many nodes
  - Short execution time means one instance would not scale anyway
  - Most impacted by difference between MTA-2 and Eldorado
- **Connected Components**
  - "Bully" algorithm is best performing on MTA-2, should be best performing on Eldorado
  - Per node performance should be comparable between MTA-2 and Eldorado

Implications for Graph Kernels

- **Subgraph Isomorphism and Sparse Matrix Vector Multiply are the poster children for "codes that should scale"**
  - Both have good "work to memory access" ratios
  - Both have Eldorado friendly global access rates
  - Both are very DRAM buffer friendly (regardless of the number of threads per processor)
  - Both could benefit from Eldorado specific optimization
- **Subgraph Isomorphism could use a "local copy of the subgraph" to shift from global to local accesses**
- **Sparse Matrix Vector Multiply could apply some distributed memory techniques to move more accesses locally**

Optimizing for Eldorado

- **Results paint a worst case scenario because the software was optimized for MTA-2**
- **Applications could become local memory aware**
  - MTA-2 had no exploitable locality, but Eldorado apps could attempt to exploit local buffer
- **Compiler optimizations could differ**
  - Register spill/fill avoided "at all cost", but cheap on Eldorado (may be able to reduce remote loads)
  - Instruction ordering could consider stack to be "closer" to increase LookAhead for remote accesses
- **Apps will need to become more hotspot aware**

Conclusions

- **Graph algorithms are demanding in terms of mem. reference**
  - The make more memory references (50-80%) and more of them go to the network (50%)
  - But, this is worst case scenario (not optimized for Eldorado)
- **Graph algorithms should still scale well**
  - Not as well as the MTA-2, but better than any other platform
  - DRAM buffer should perform well under this usage model
  - Network performance is within a factor of 2 or 3 of "enough"
  - Hotspots are bad, but not as bad as they could have been
- **Eldorado will be the fastest graph machine available**