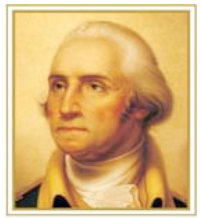


THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON DC

HPCL
The George Washington University
WASHINGTON DC
High Performance Computing Lab

Arctic Region Supercomputing Center

DNA and Protein Sequence Alignment with High Performance Reconfigurable Systems

Mohamed Abouellail, Esam El-Araby, Mohamed Taher,
and Tarek El-Ghazawi, George Washington University
Gregory B. Newby, Arctic Region Supercomputing Center

Presentation at AHS-2007, Edinburgh. Wednesday August 8 2007

Outline

- A word from your presenters…
- Introduction
- Implementation Approach
- Testbeds
- Experimental Results
- Conclusions

Other group activities

- The research team is heavily involved with high-performance reconfigurable computing, evaluation of new hardware (such as multicore CPUs), as well as associated languages and tools
- GWU is co-host of CHREC http://www.chrec.ufl.edu/ ; ARSC is a charter member
- Recent publications include an analysis of high level languages for FPGAs
- Next week: multicore symposium (accessible via AccessGrid). See www.arsc.edu

String Matching is the basis for sequence alignment
(Introduction)

- **String Matching**
  - **Detecting the occurrence of a particular substring, called the** pattern**, in another string, called the** text
- **Types of String matching:**
  - **Exact string matching**
  - **Approximate string matching**
- **Exact string matching:**
  - **Involves match patterns, where they exist completely, that is unbroken and with no irrelevant data in between any letters**
  - **Numerous Applications : NIDS, text editing, …etc.**
- **Approximate string matching:**
  - **Pattern rarely matches the text completely**
  - **Finds application in Computational biology (DNA sequence alignment), image detection, handwriting recognition…etc.**

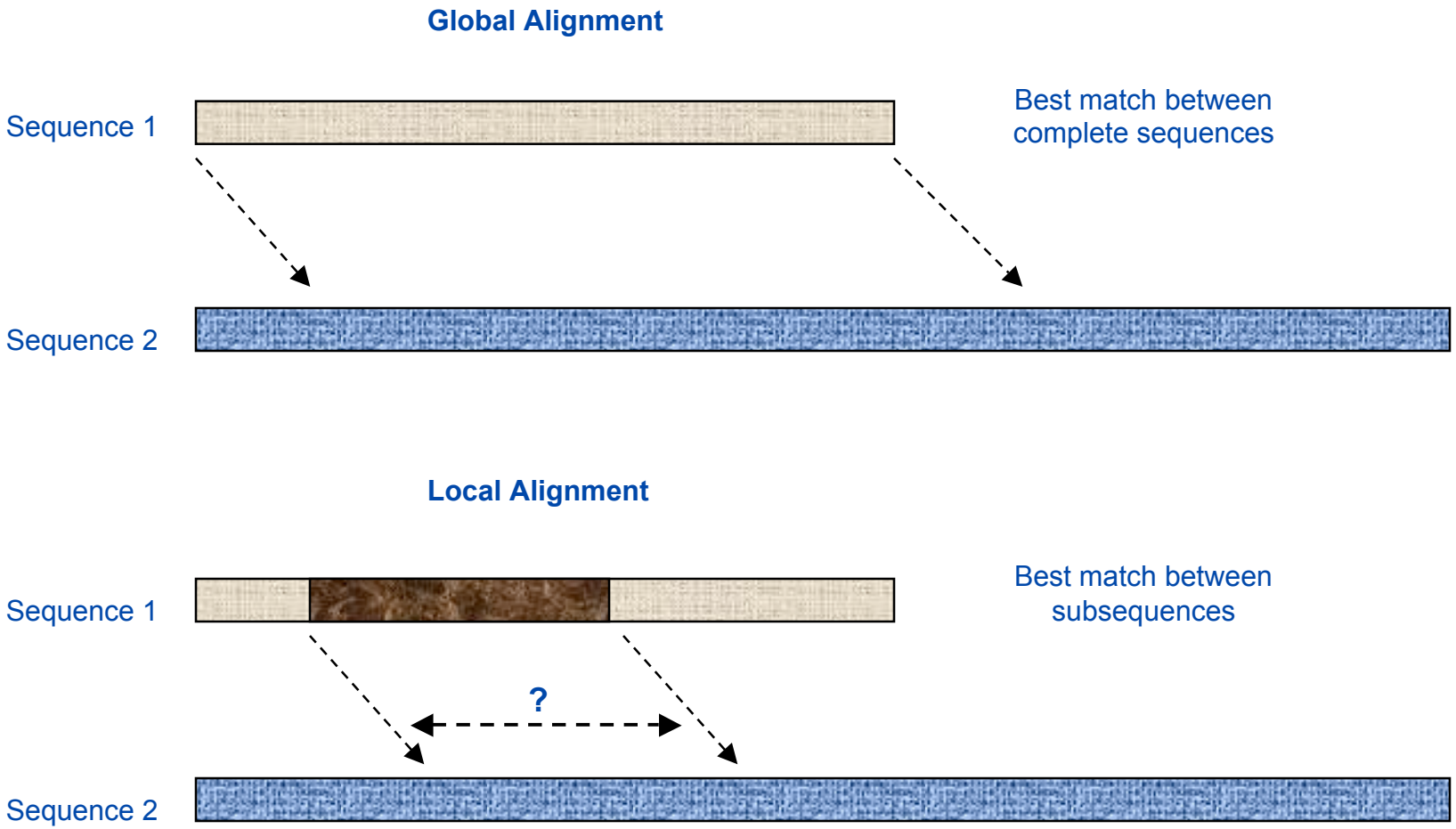
Sequence Alignment
(Smith-Waterman Algorithm)

- **Why align two protein or DNA sequences?**
  - Determine whether they are descended from a common ancestor (homologous)
  - Infer a common function
  - Locate functional elements
  - Infer protein structure, if the structure of one of the sequences is known
- **S-W genomic comparison and alignment algorithm**
  - Similar to BLAST, but 10x slower
  - Provably optimum- the "gold standard" for alignment algorithms
  - Based on Dynamic Programming
- **Two-step process**
  - Create scoring matrix and find maximum score
    - "forward pass"
  - Work back to determine alignment
    - "traceback"

Sequence Alignment Algorithms

- **Dynamic Programming**
  - Break large problems into smaller, simpler sub problems
  - Solve sub problems optimally and recursively
  - Use these optimal solutions to construct an optimal solution for the original problem
- **The Smith-Waterman algorithm**
  - Implements the dynamic programming technique
  - Performs local sequence alignment; that is, for determining similar *regions* between two nucleotide or protein sequences

Global vs. Local Alignment



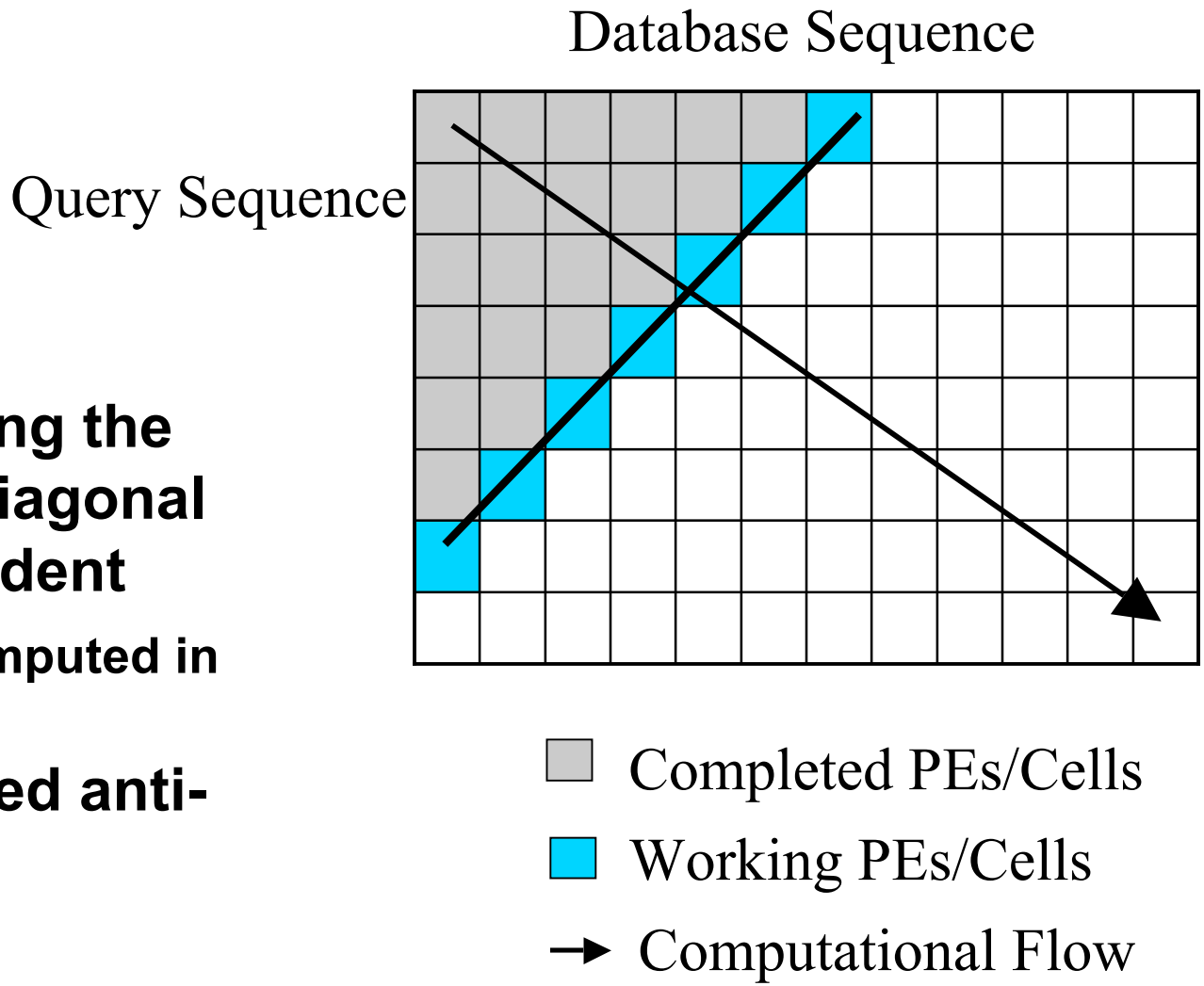
Outline

- Introduction
- **Implementation Approach**
- Testbeds
- Experimental Results
- Conclusions

Implementation for Hardware

- **Cellular Automata Approach**
  - Matrix elements are identical PEs/Cells
  - Cells communicate with their neighbors updating the local scores and propagating the maximum local score
  - Maximum score found in the last cell calculated
- **Virtualization & Scheduling**
  - Using sliding window to traverse entire virtual scoring matrix
  - Last column of every iteration is fed back to the first column in the following iteration

Anti-Diagonal Wave-front Data Dependency



- All cells along the same anti-diagonal are independent
  - Can be computed in parallel
- Matrix is filled anti-diagonally

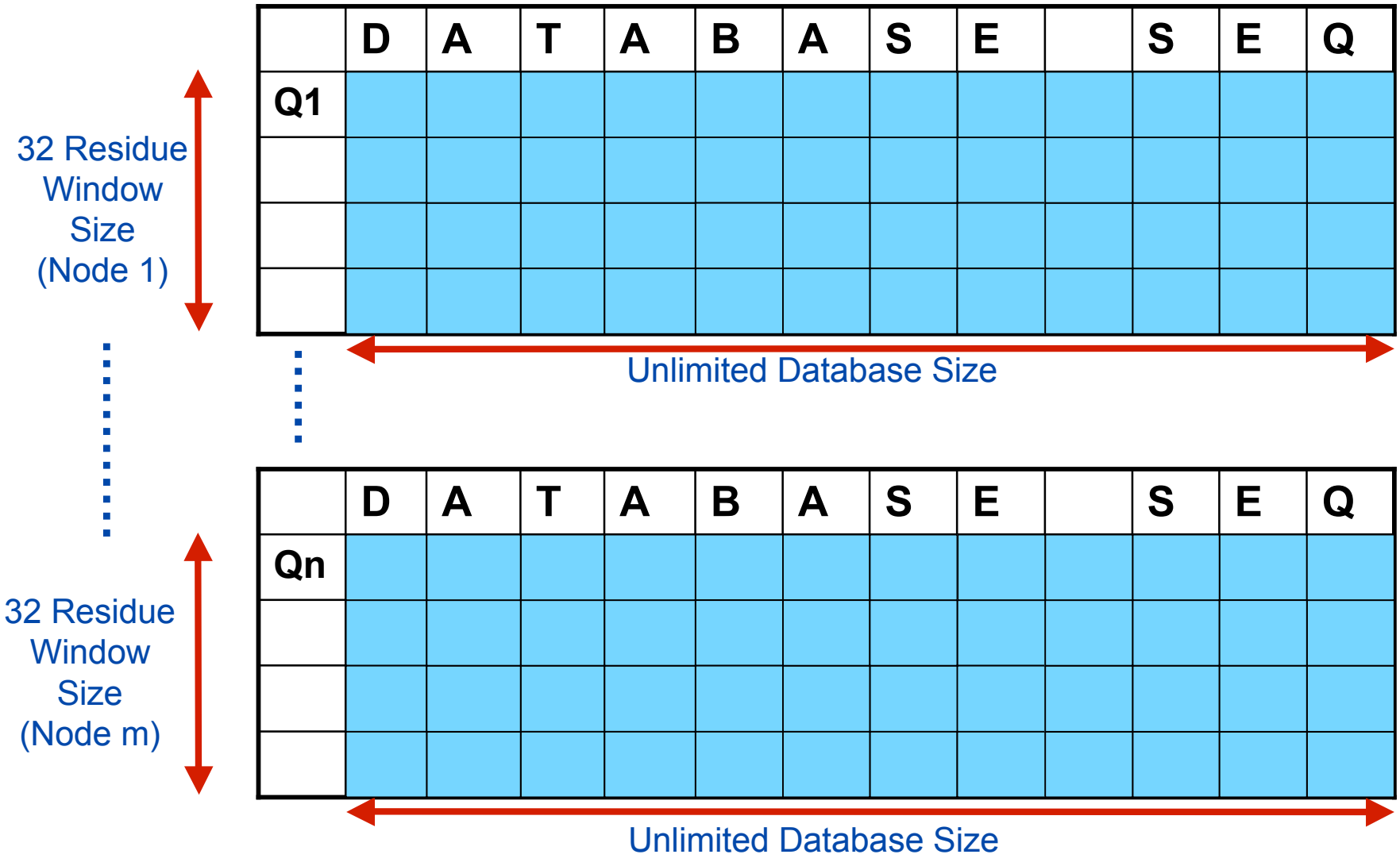
Computing the Similarity Matrix

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + s(x_i, y_j) \\ F(i-1,j) + gap\_penalty \\ F(i,j-1) + gap\_penalty \\ 0 \end{cases}$$

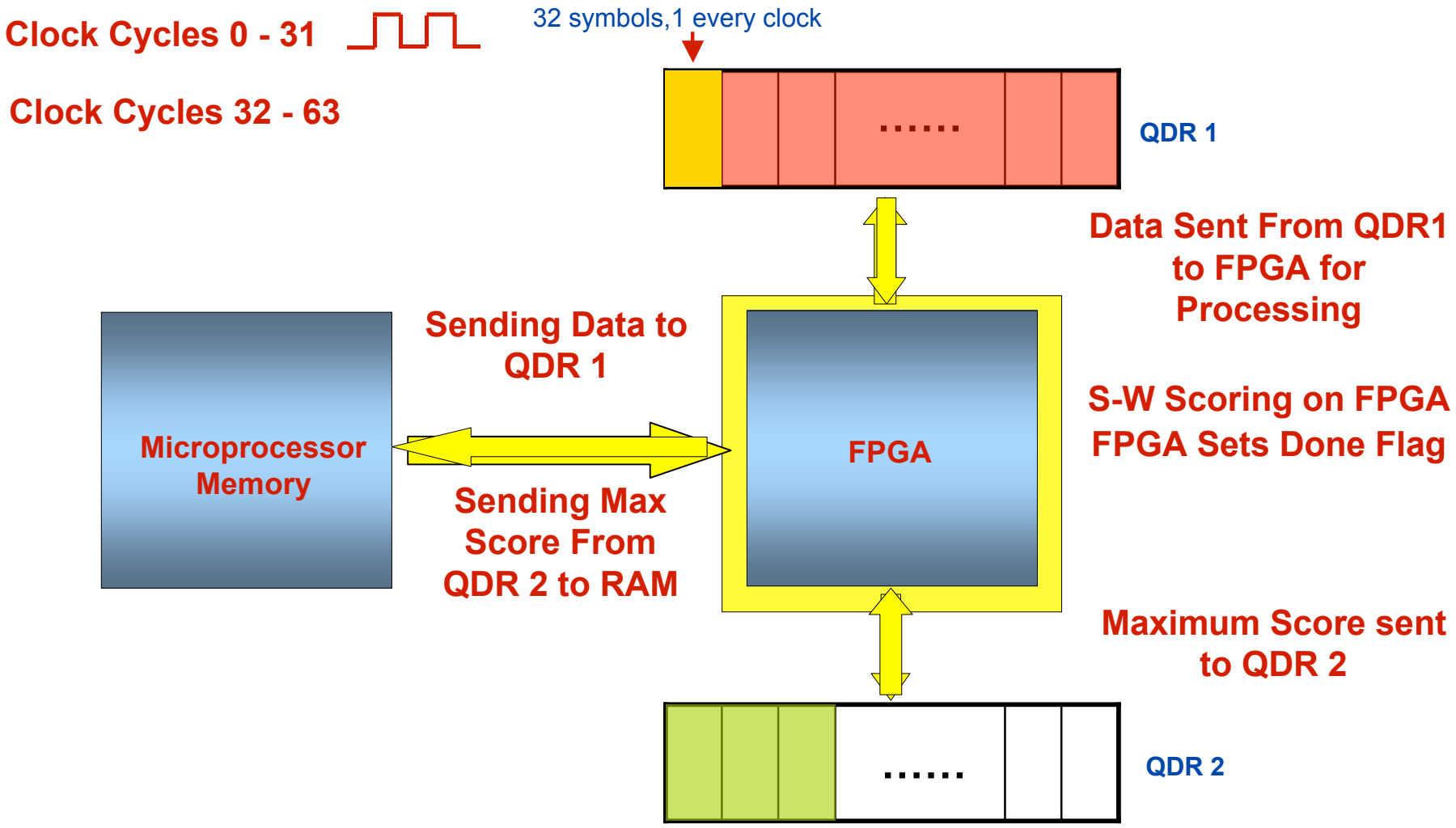
	-	T	T	T	C	A	T	A	G
-	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	0	0	5
G	0	0	0	0	0	0	0	0	5
T	0	5	5	5	0	0	5	0	0
T	0	5	10	10	1	0	5	1	0
A	0	0	1	6	6	6	0	10	0
T	0	5	5	6	2	2	11	0	6
G	0	0	1	1	2	0	0	7	5
C	0	0	0	0	6	0	0	0	3

Implementation for Hardware (cnt'd)

(32x1 Sliding Window)

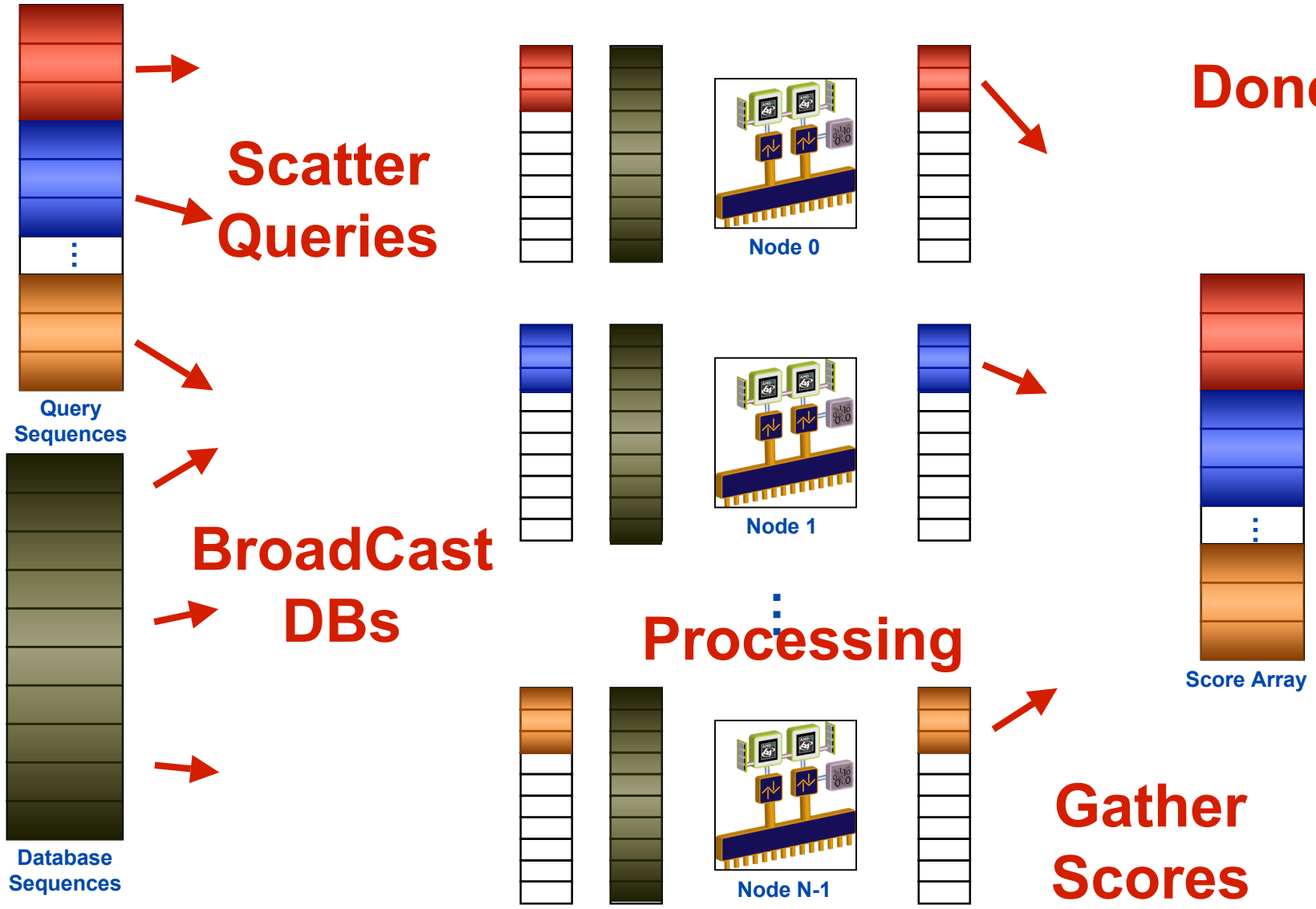


Data Transfers Scenario



Implementation for Hardware (cnt'd)

(MPI Implementation)



Outline

- Introduction
- Implementation Approach
- **Testbeds**
- Experimental Results
- Conclusions

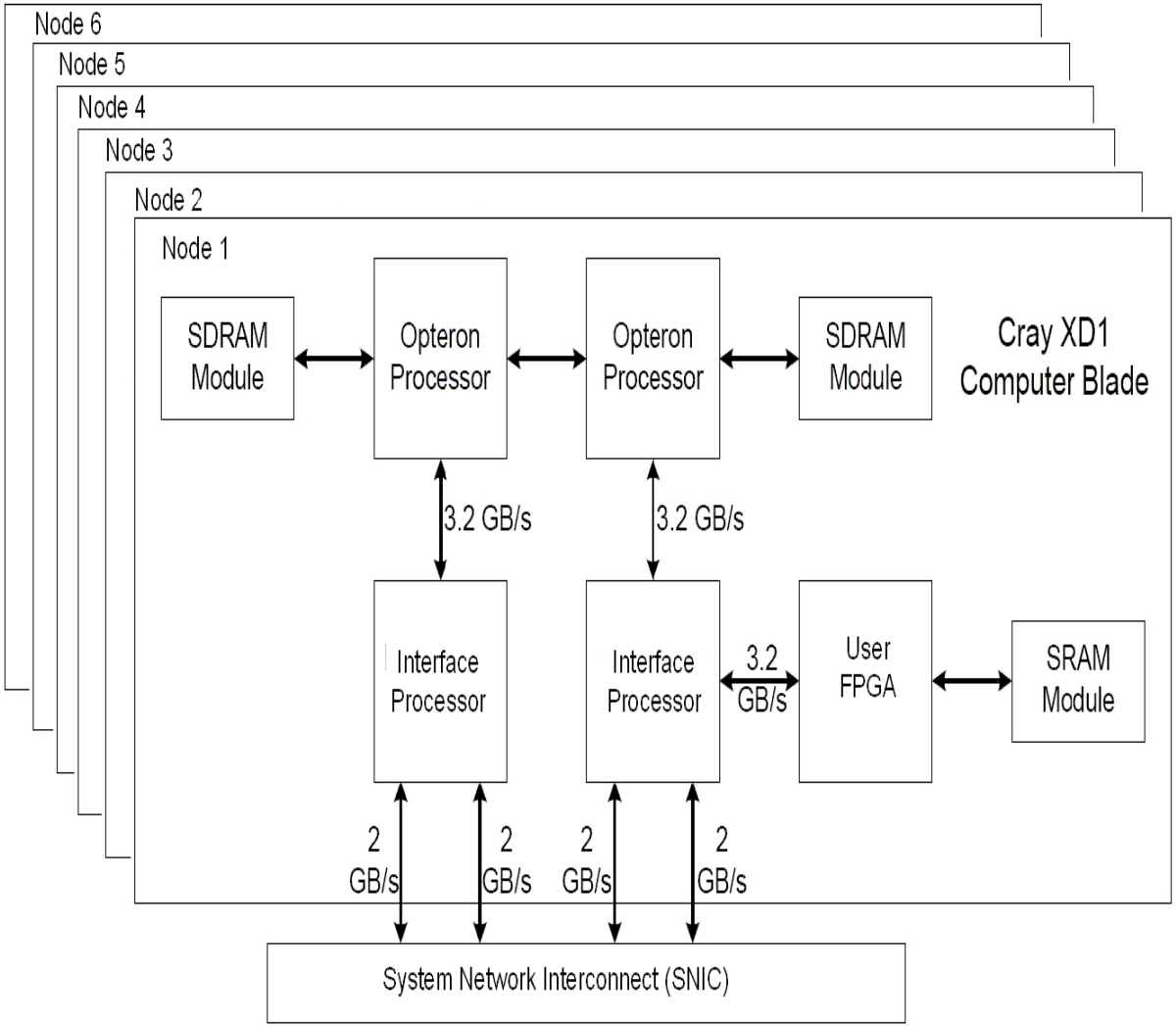
Cray XD1 System Architecture
(Six Chassis)

Compute
- 12 AMD Opteron 32/64 bit, x86 processors
- High Performance Linux

RapidArray Interconnect
- 12 communications processors
- 1 Tb/s switch fabric

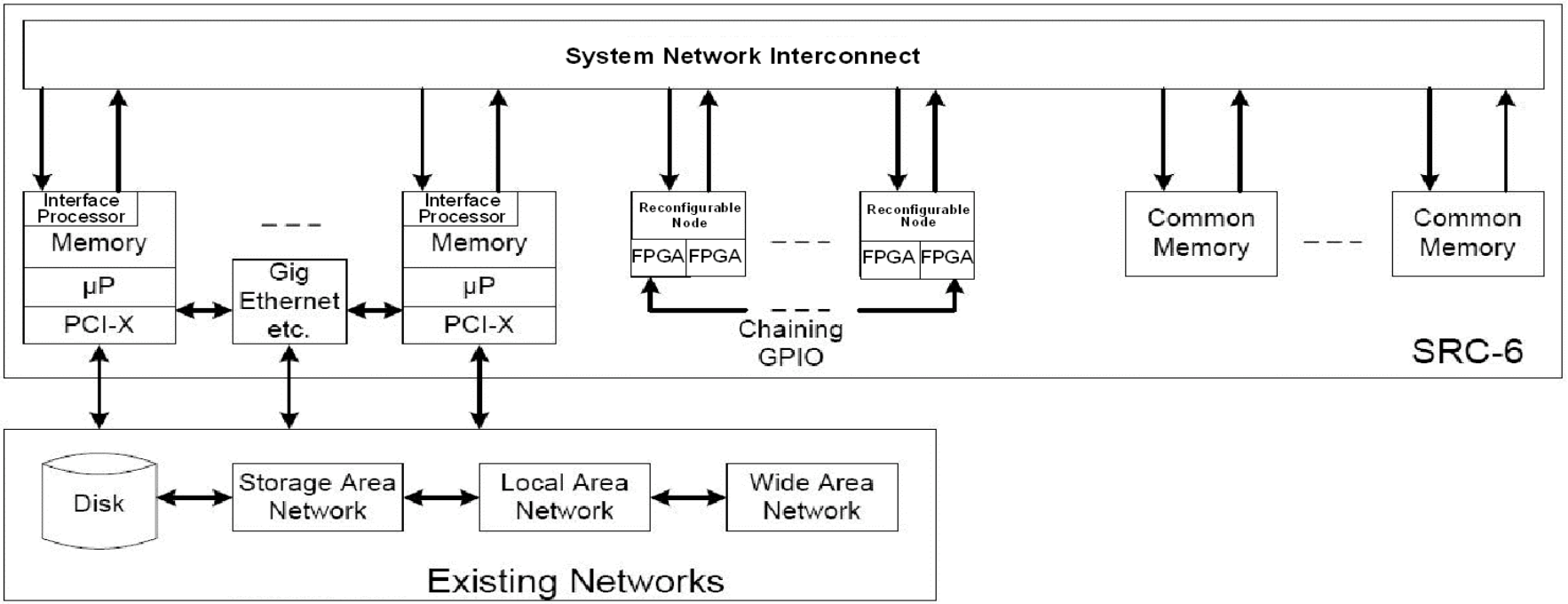
Active Management
- Dedicated processor

Application Acceleration
- 6 co-processors



SRC Hi-Bar™ Based Systems

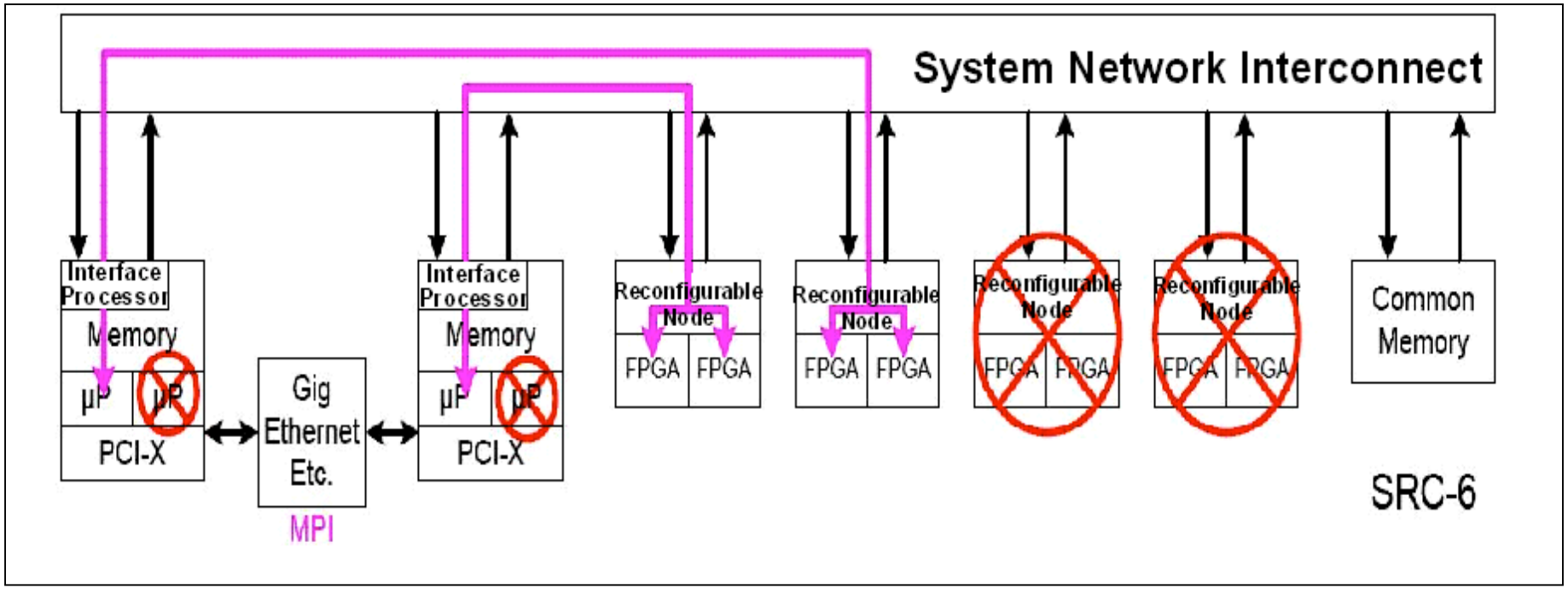
- System Network Interconnect (Hi-Bar) sustains 1.4 GB/s per port with 180 ns latency per tier
- Up to 256 input and 256 output ports with two tiers of switch
- Common Memory (CM) has controller with DMA capability
- Controller can perform other functions such as scatter/gather
- Up to 8 GB DDR SDRAM supported per CM node



Outline

- Introduction
- Implementation Approach
- Testbeds
- **Experimental Results**
- Conclusions

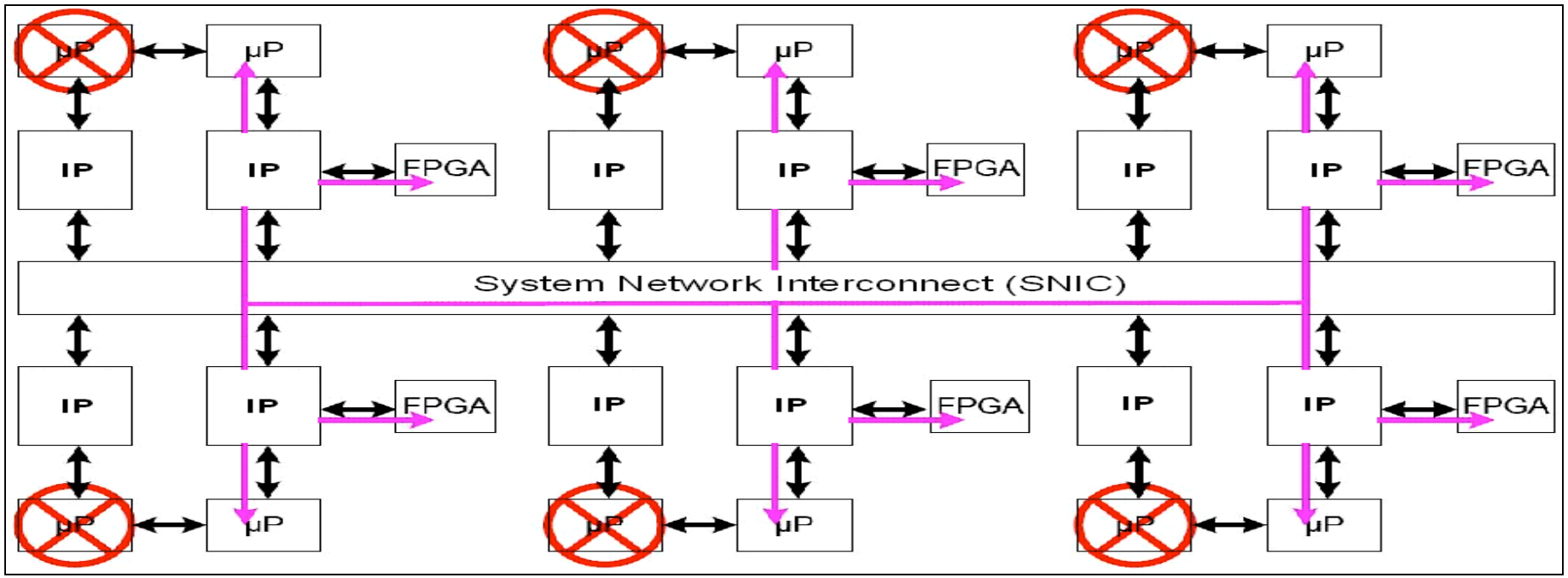
MPI Utilization on SRC-6



⊗ Unutilized uP/FPGA
uP: Microprocessor

- Network Interface Cards cannot be efficiently shared
  - Only two MPI processes were implemented

MPI Utilization on Cray-XD1



⊗ Unutilized uP/FPGA
IP: Interface Processor
uP: Microprocessor (Opteron)

- All Nodes were exploited using MPI
  - However, only one of the two microprocessors on each node sufficed

Performance Results

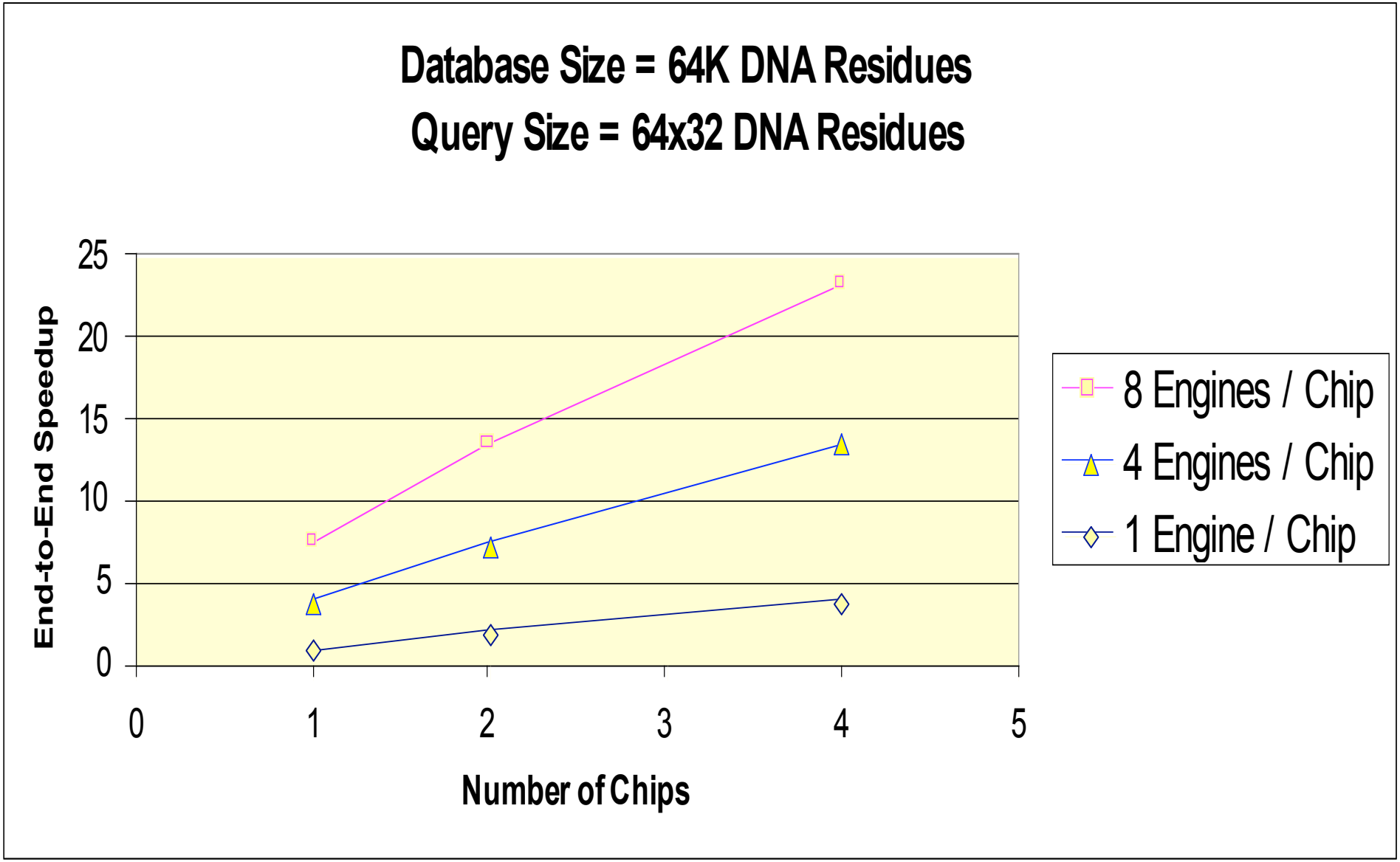
				Expected		Measured	
				Throughput (GCUPS)	Speedup	Throughput (GCUPS)	Speedup
FASTA SSEARCH34	Opteron 2.4GHz	DNA		NA	NA	0.065	1
		Protein		NA	NA	0.130	1
GWU	SRC 100 MHz (32x1)	DNA	1 Engine/Chip	3.2	49.2	3.19 → 12.2 1 →4 Chips	49 → 188 1→4 Chips
			4 Engines/Chip	12.8	197	12.4 → 42.7 1 →4 Chips	191 → 656 1→4 Chips
			8 Engines/Chip	25.6	394	24.1 → 74 1→ 4 Chips	371 → 1138 1→4 Chips
		Protein		3.2	24.6	3.12 → 11.7 1 →4 Chips	24 → 90 1→4 Chips
	XD1 200 MHz (32x1)	DNA	1 Engine/Chip	6.4	98	5.9 → 32 MPI 1→6 nodes	91 → 492 MPI 1→6 nodes
			4 Engines/Chip	25.6	394	23.3 → 120.7 MPI 1→6 nodes	359 → 1857 MPI 1→6 nodes
			8 Engines/Chip	51.2	788	45.2 → 181.6 MPI 1→6 nodes	695 → 2794 MPI 1→6 nodes
		Protein		6.4	49	5.9 → 34 MPI 1→6 nodes	45 → 262 MPI 1→6 nodes

SRC-6

DNA

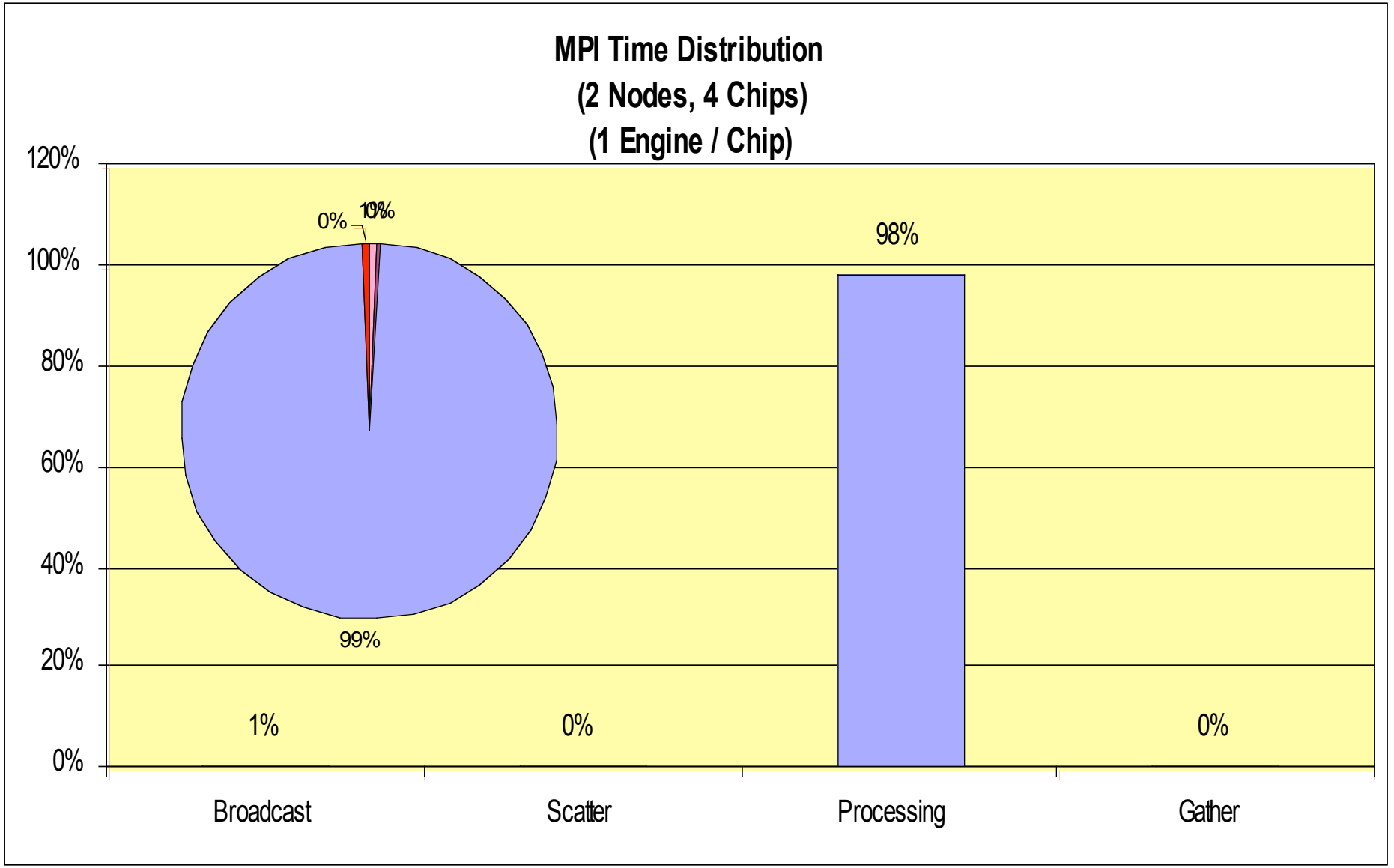
Smith-Waterman Scalability on SRC-6

(window of 32x1 DNA residues)



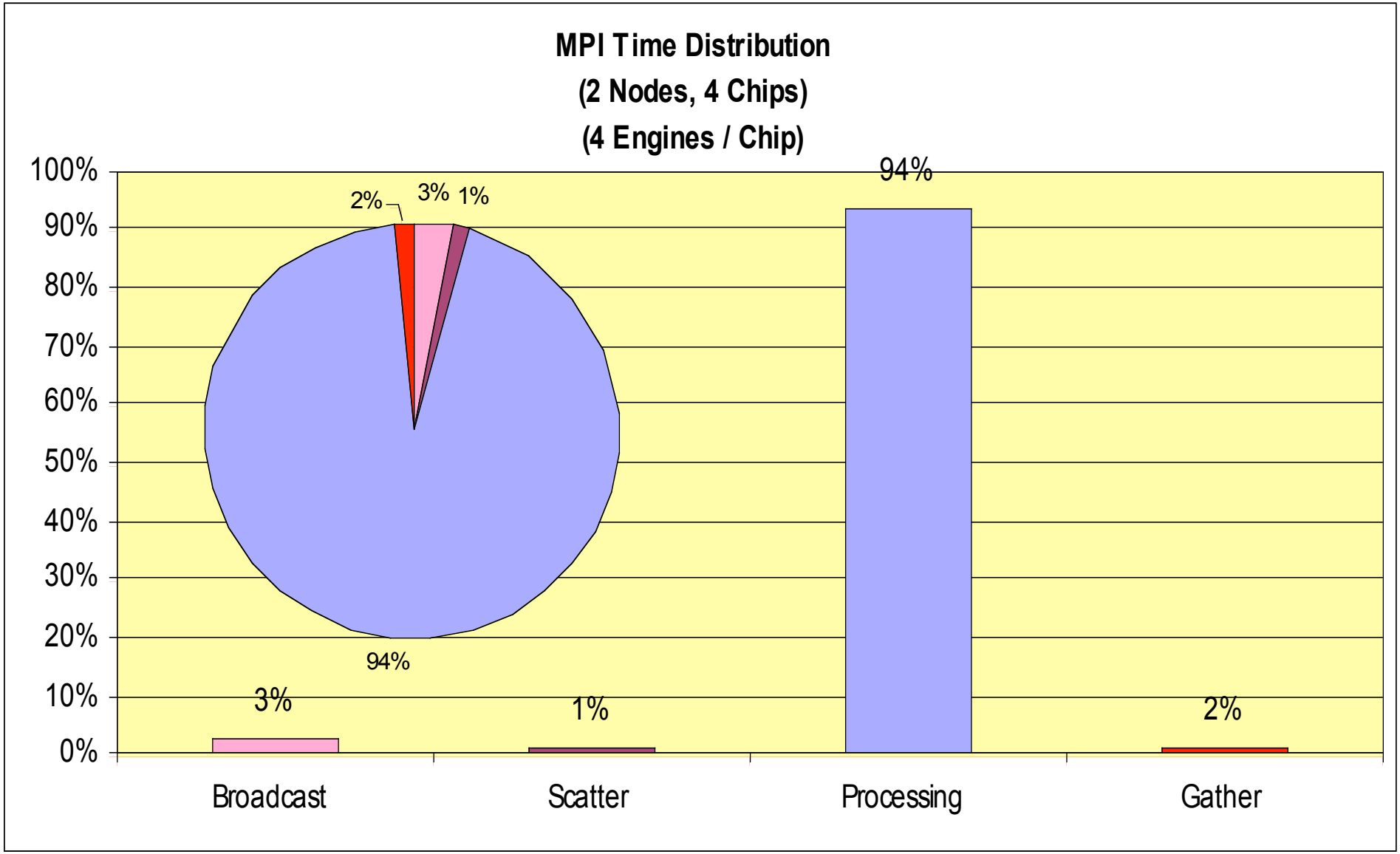
Time Distribution of Smith-Waterman on SRC-6

(window of 32x1 DNA residues)



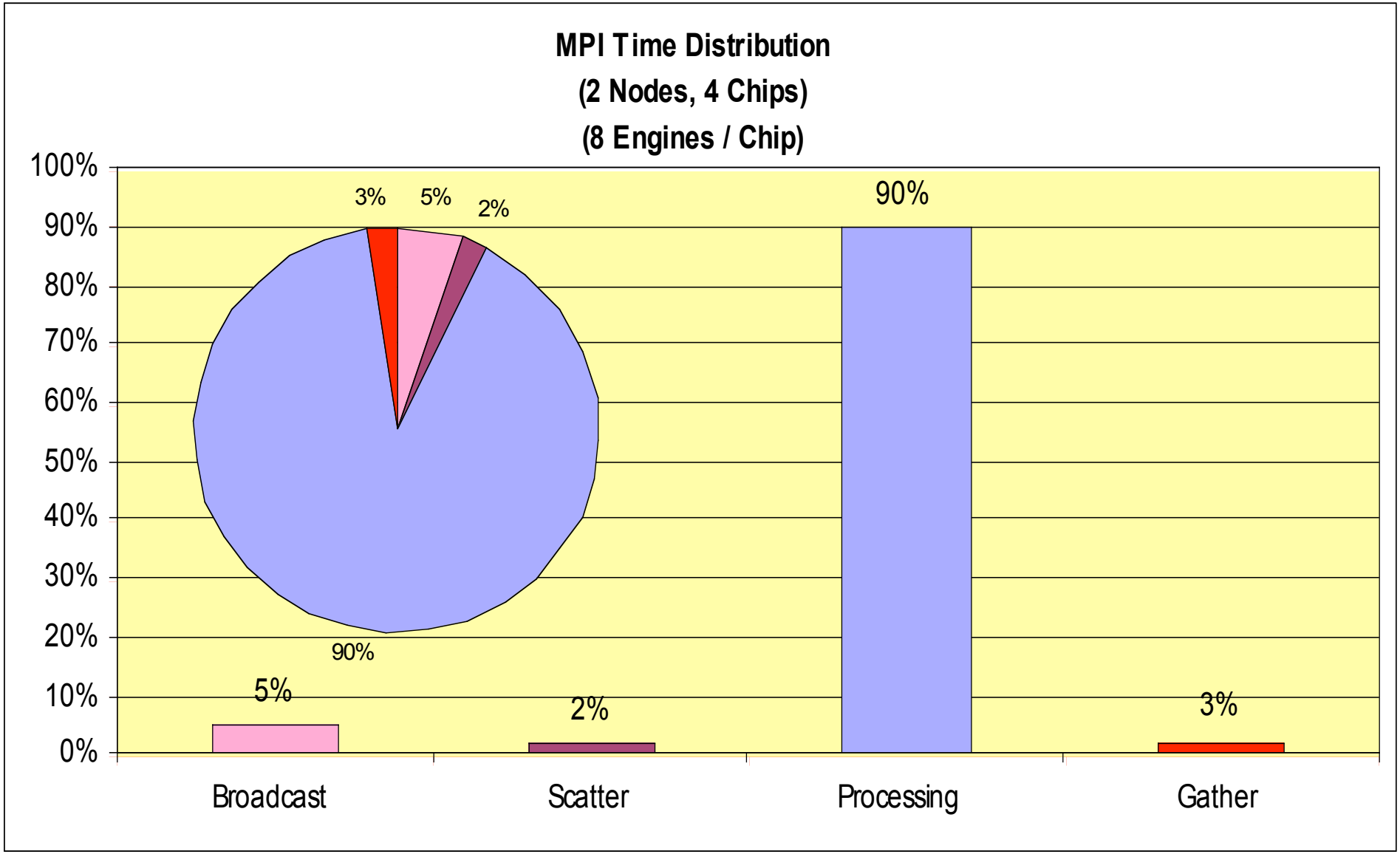
Time Distribution of Smith-Waterman on SRC-6

(window of 32x1 DNA residues)



Time Distribution of Smith-Waterman on SRC-6

(window of 32x1 DNA residues)

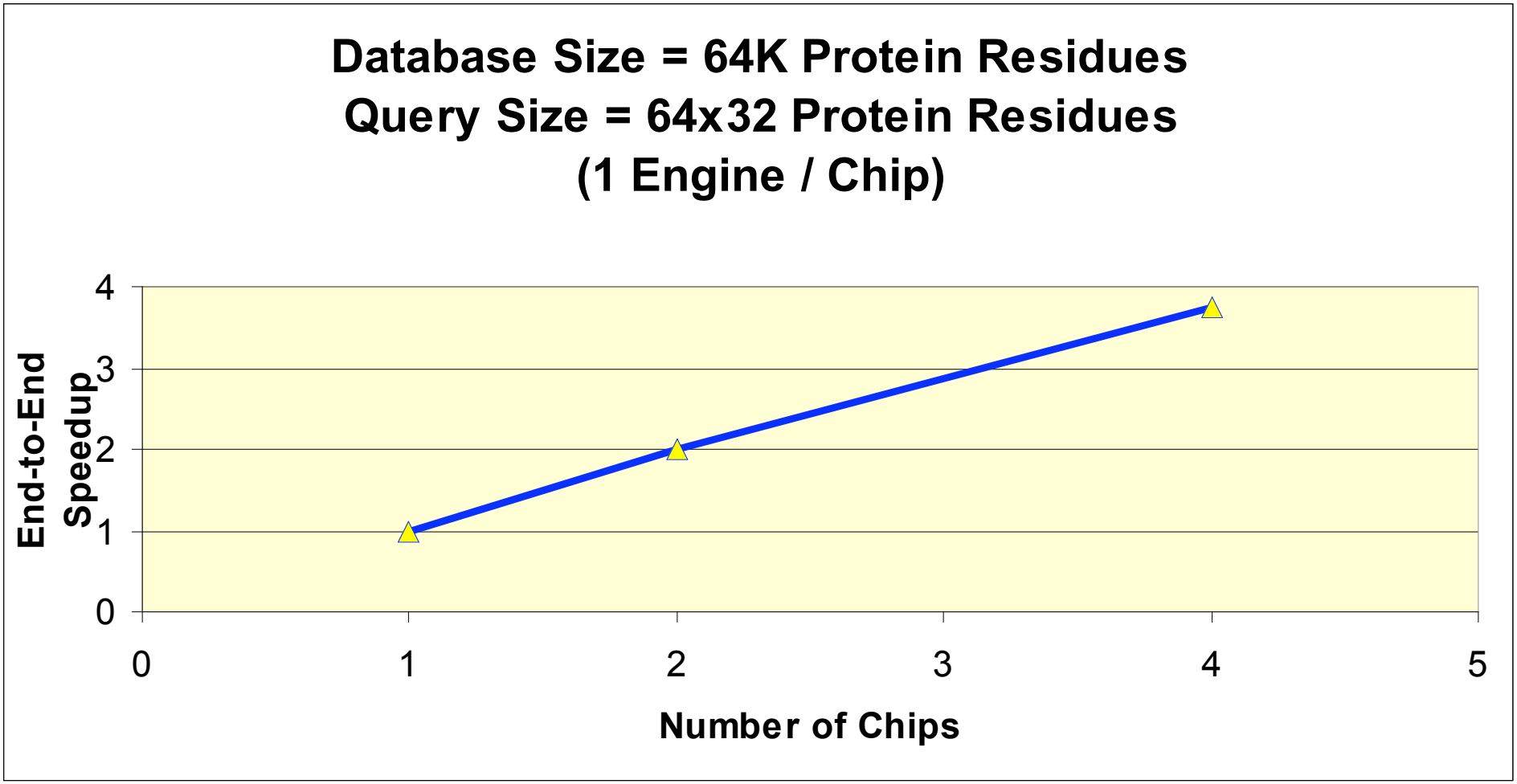


SRC-6

Protein

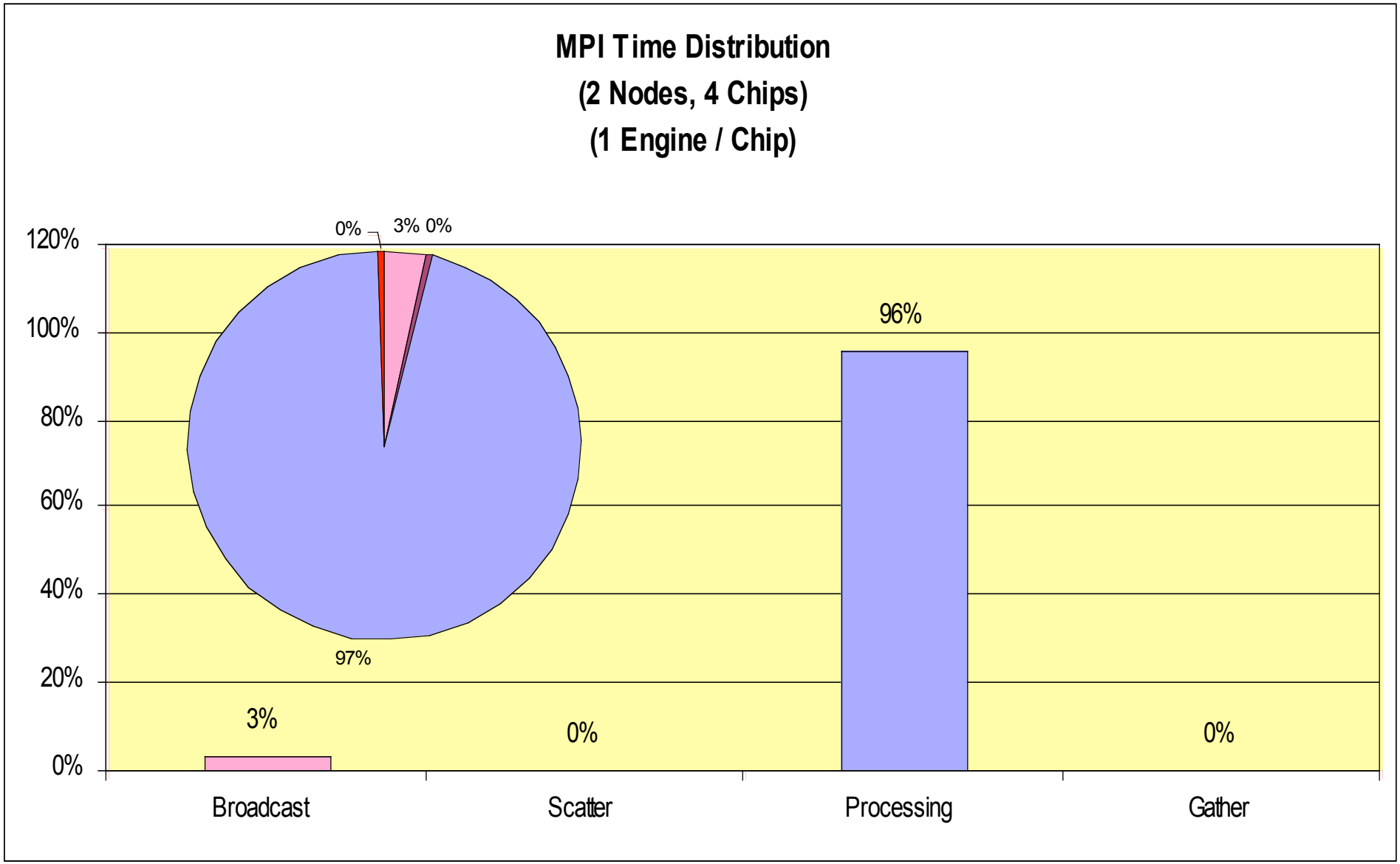
Smith-Waterman Scalability on SRC-6

(window of 32x1 Protein residues)



Time Distribution of Smith-Waterman on SRC-6

(window of 32x1 Protein residues)

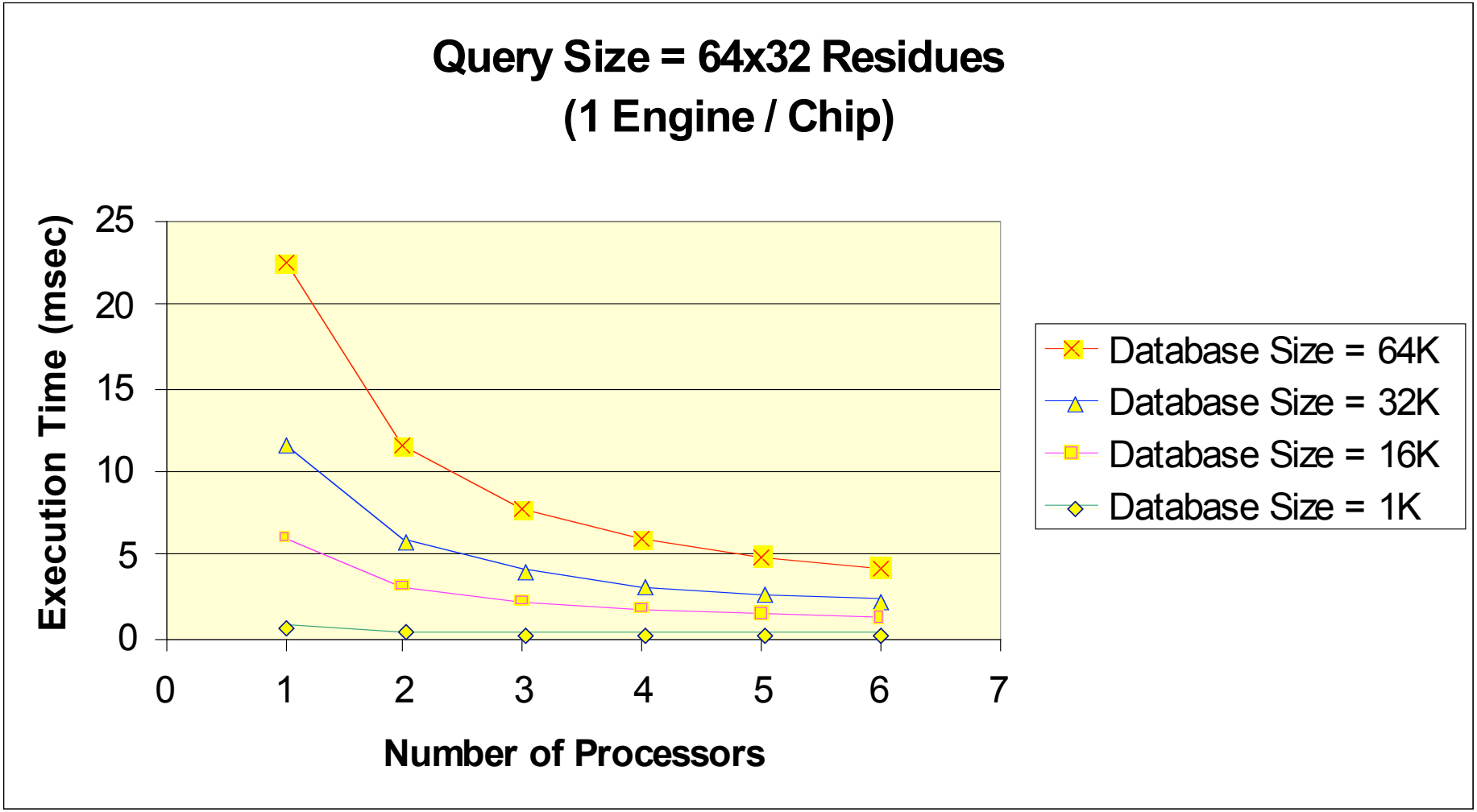


Cray-XD1

DNA

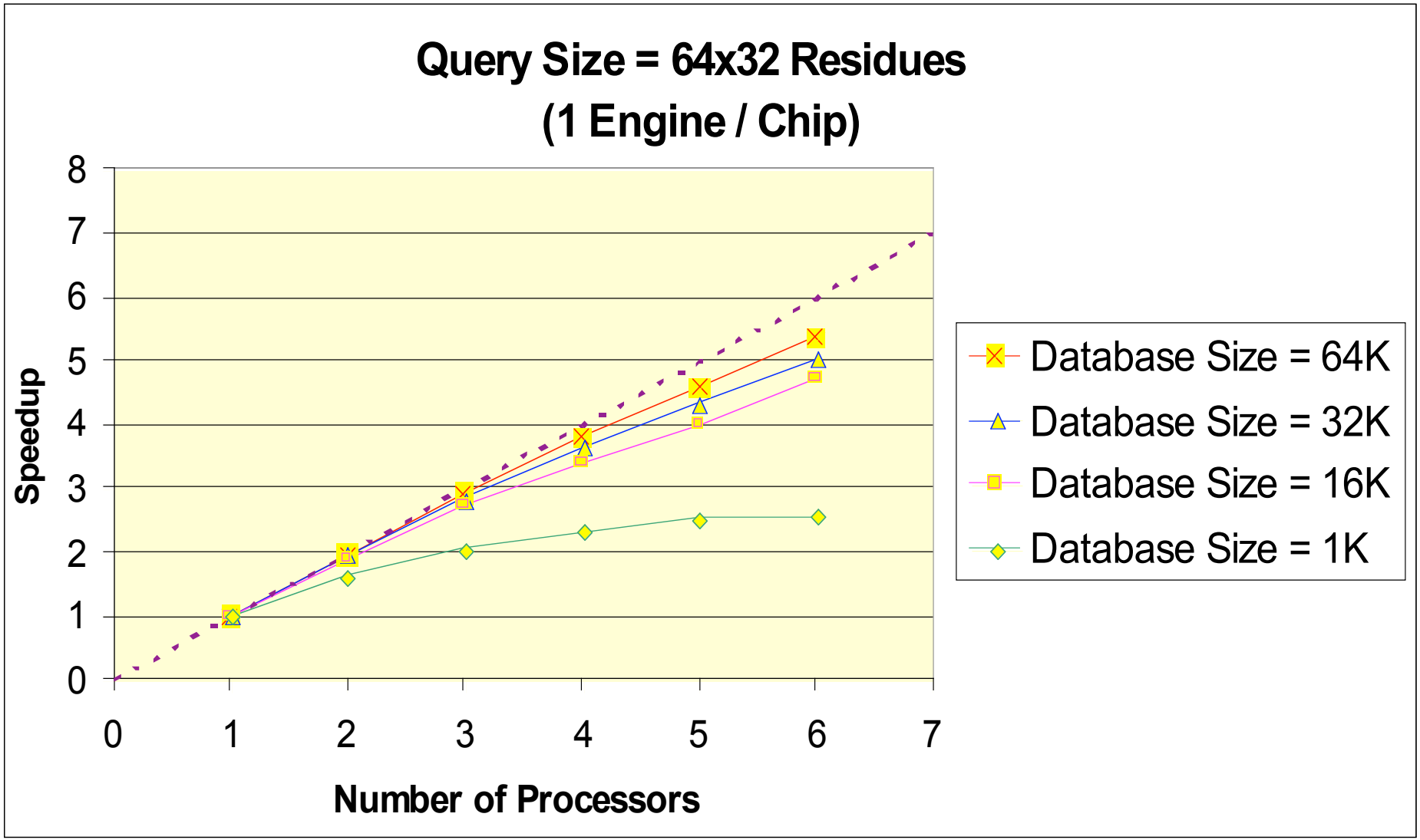
Smith-Waterman Scalability on XD1

(window of 32x1 DNA residues)



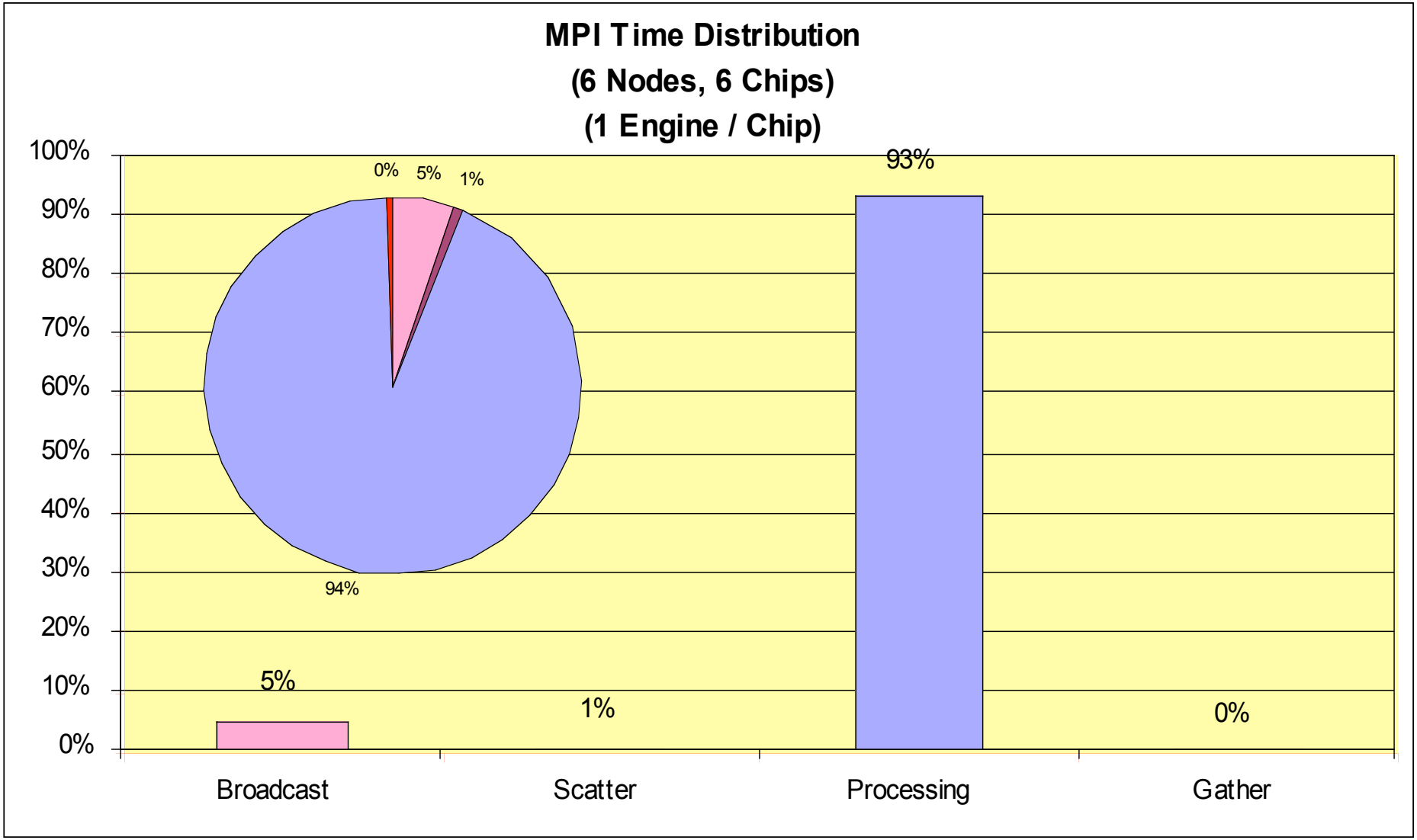
Smith-Waterman Scalability on XD1

(window of 32x1 DNA residues)

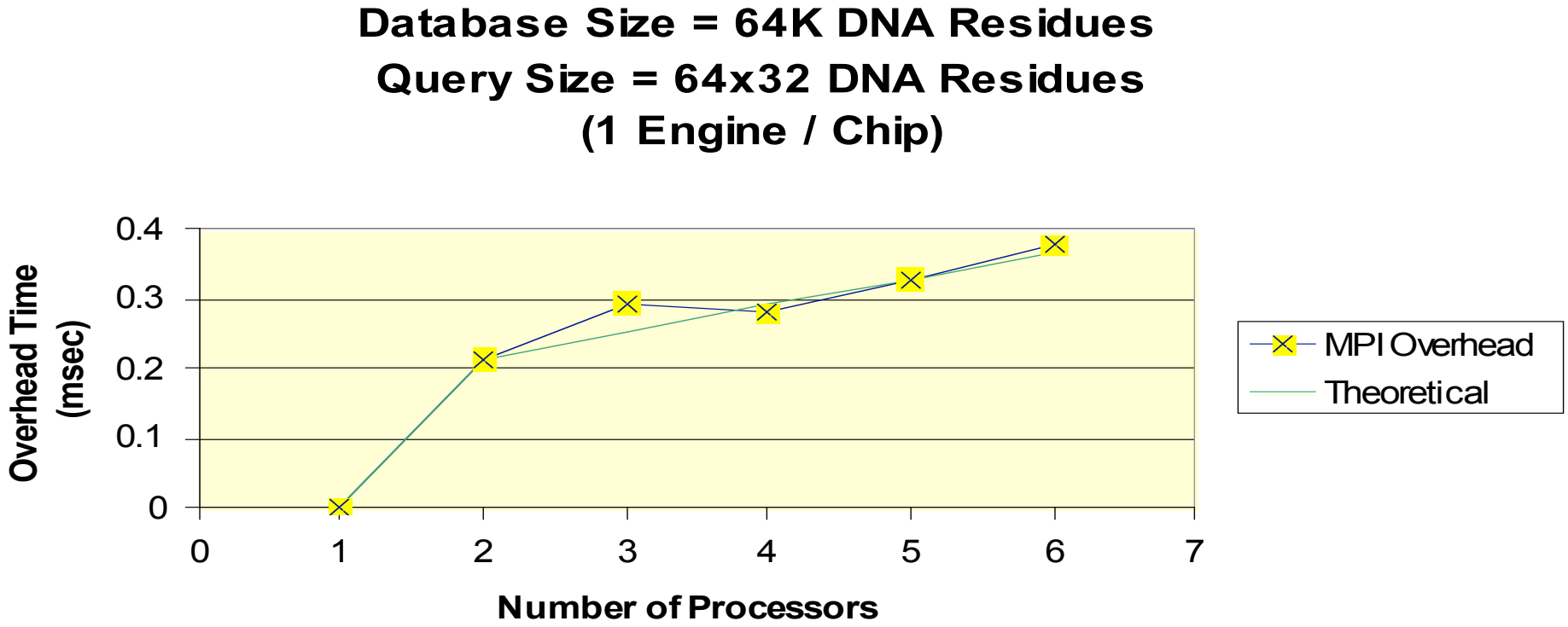


Time Distribution of Smith-Waterman on XD1

(window of 32x1 DNA residues)



MPI Overhead and Computation Speedup
(window of 32x1 DNA residues)



$$t_{MPI}^{theoretical}(N) = \begin{cases} 0 & N = 1 \\ t_{MPI}^{measured}(2) + \alpha_{MPI}(N-2) & N > 1 \end{cases}$$

where $N \equiv$ *number of processors,*

$$\text{and } \alpha_{MPI} = \frac{\sum_{N=2}^{6}(N-2)\left(t_{MPI}^{measured}(N) - t_{MPI}^{measured}(2)\right)}{\sum_{N=2}^{6}(N-2)^2}$$

$$t_{comp}^{theoretical}(N) \equiv \frac{t_{comp}(1)}{S_{comp}(N)}$$

$$S_{comp}(N) = 1 + \alpha_{comp}(N-1)$$

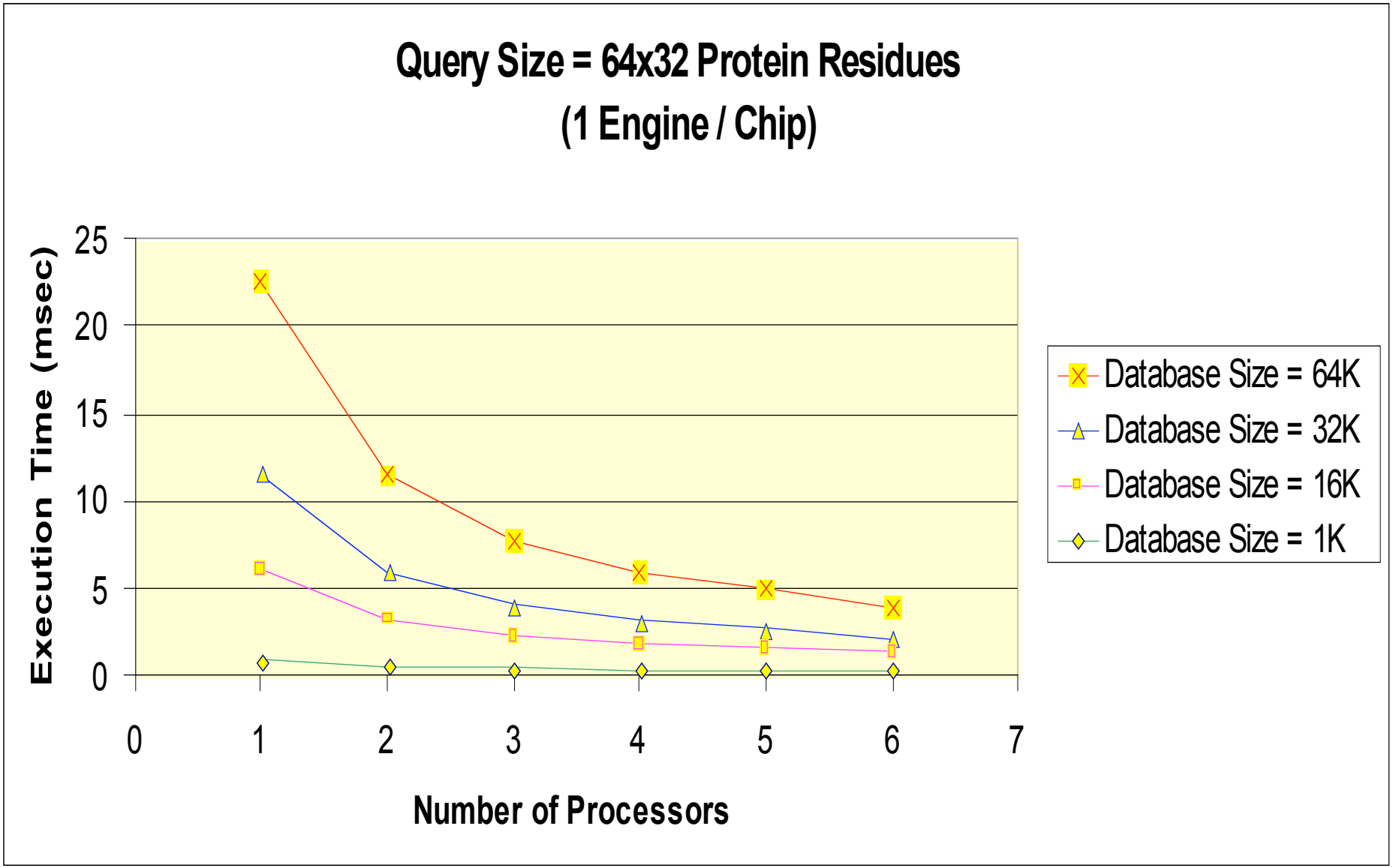
$$\text{where } \alpha_{comp} = \frac{\sum_{N=1}^{6}(N-1)\left(\frac{t_{comp}^{measured}(1)}{t_{comp}^{measured}(N)} - 1\right)}{\sum_{N=1}^{6}(N-1)^2}$$

Cray-XD1

Protein

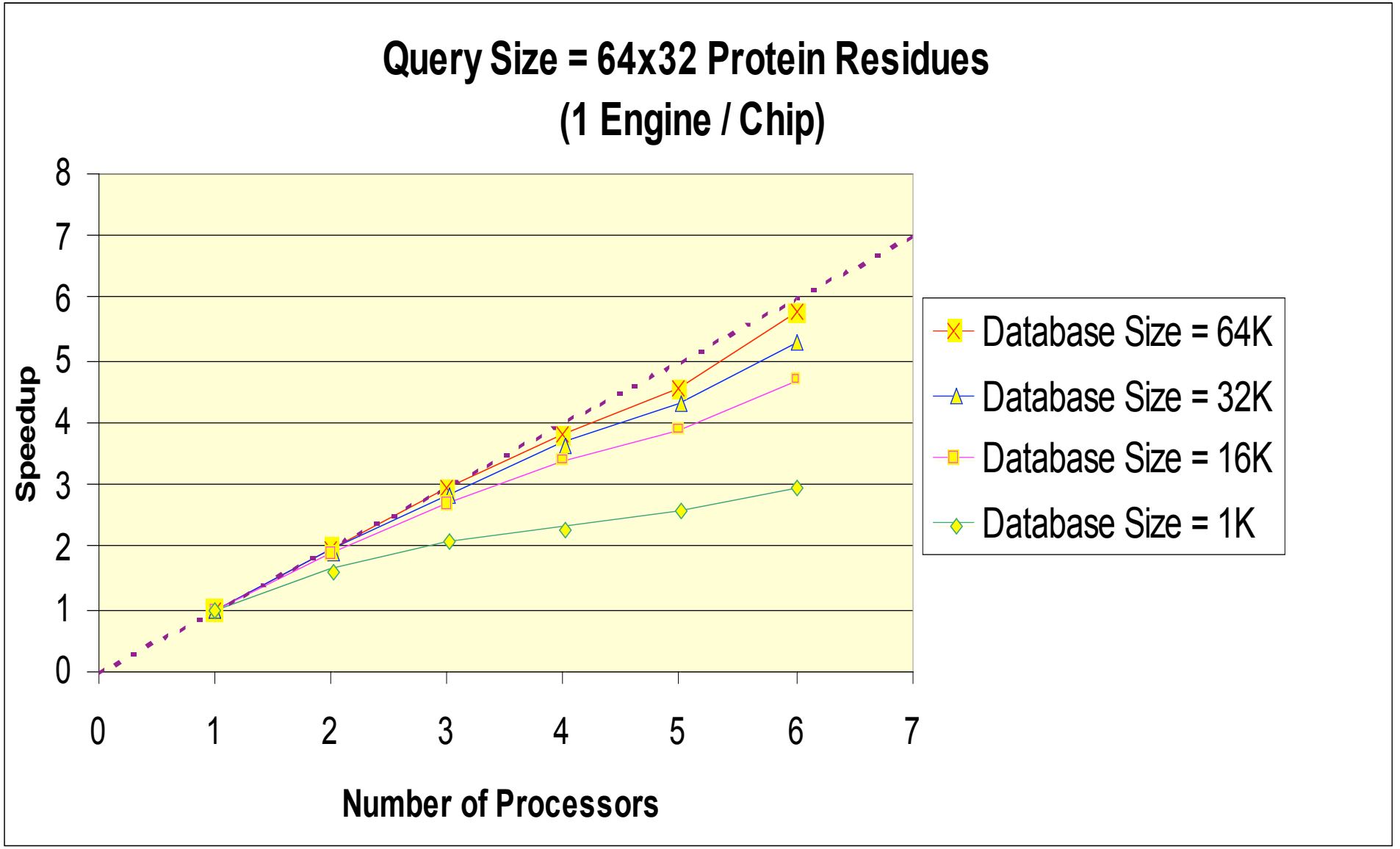
Smith-Waterman Scalability on XD1

(window of 32x1 Protein residues)



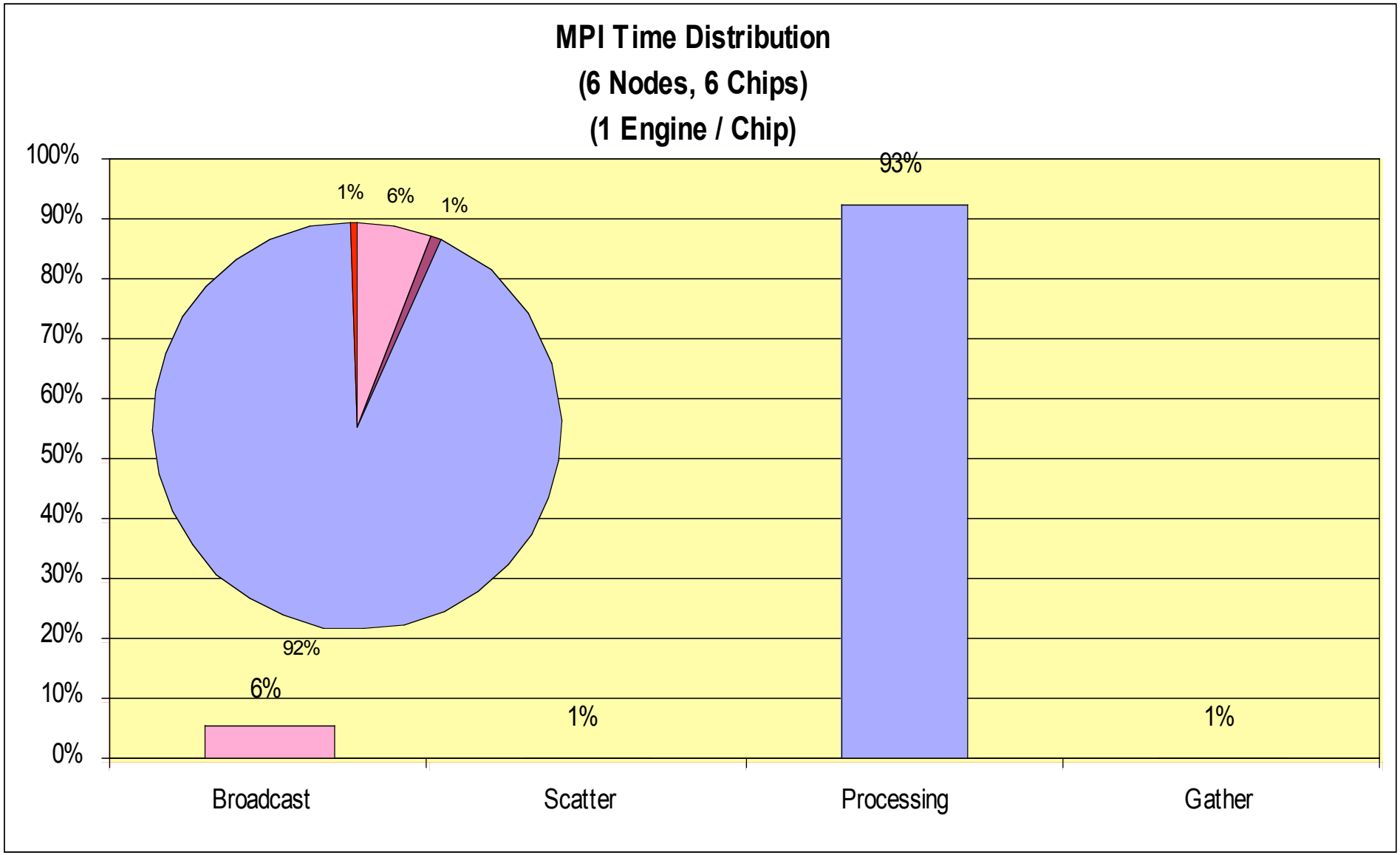
Smith-Waterman Scalability on XD1

(window of 32x1 Protein residues)

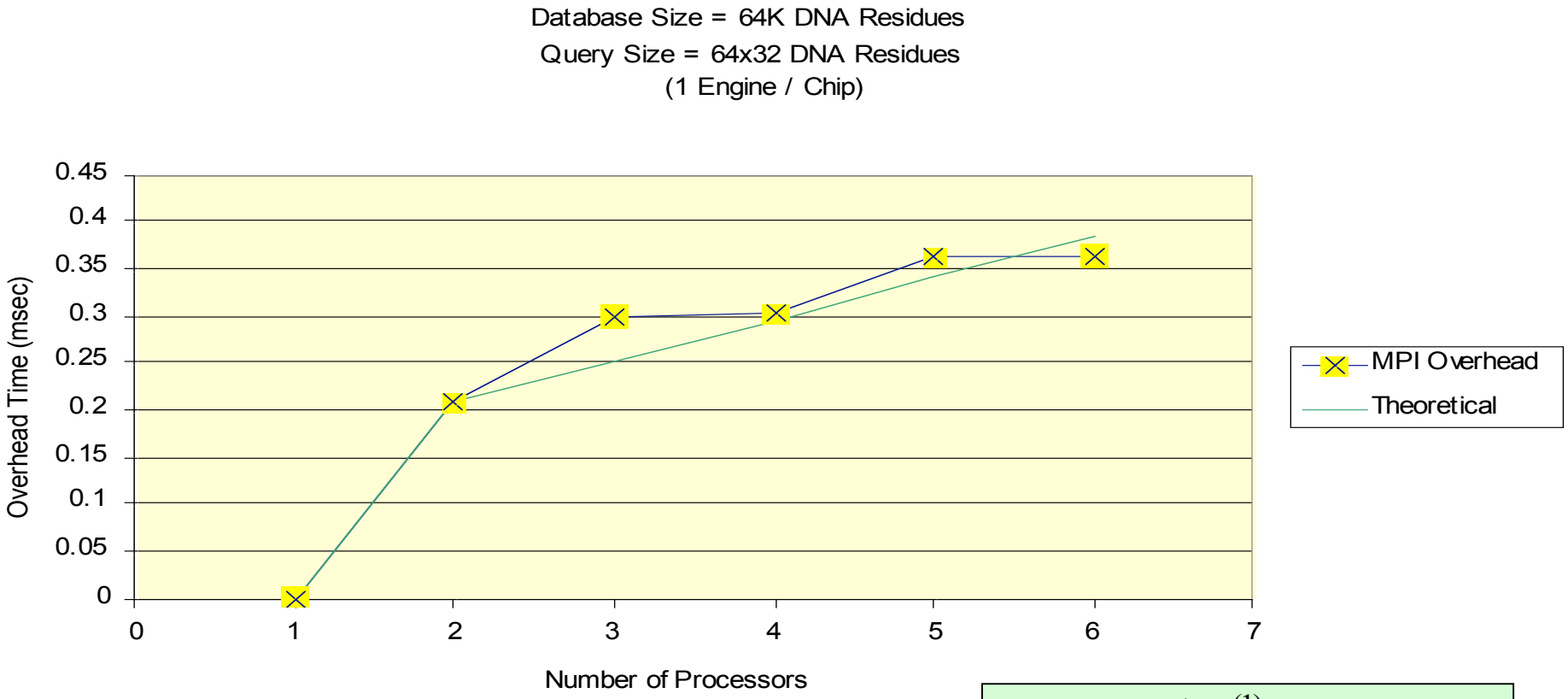


Time Distribution of Smith-Waterman on XD1

(window of 32x1 Protein residues)



MPI Overhead and Computation Speedup
(window of 32x1 Protein residues)



$$t_{MPI}^{theoretical}(N) = \begin{cases} 0 & N = 1 \\ t_{MPI}^{measured}(2) + \alpha_{MPI}(N-2) & N > 1 \end{cases}$$

where $N \equiv$ *number of processors,*

$$\text{and } \alpha_{MPI} = \frac{\sum_{N=2}^{6}(N-2)\left(t_{MPI}^{measured}(N) - t_{MPI}^{measured}(2)\right)}{\sum_{N=2}^{6}(N-2)^2}$$

$$t_{comp}^{theoretical}(N) \equiv \frac{t_{comp}(1)}{S_{comp}(N)}$$

$$S_{comp}(N) = 1 + \alpha_{comp}(N-1)$$

$$\text{where } \alpha_{comp} = \frac{\sum_{N=1}^{6}(N-1)\left(\frac{t_{comp}^{measured}(1)}{t_{comp}^{measured}(N)} - 1\right)}{\sum_{N=1}^{6}(N-1)^2}$$

Savings of HPRC
(Based on SRC-6)

Application	Speedup	SAVINGS		
		Cost Savings	Power Savings	Size Reduction
Smith-Waterman (DNA Sequencing)	1138	6x	313x	34x

- **Assumptions**
  - **100% cluster efficiency**
  - **Cost Factor μP : RP → 1 : 200**
  - **Power Factor μP : RP → 1 : 3.64**
    - Reconfigurable processor (based on SRC-6): 200 W
    - μP board (with two μPs): 220 W
  - **Size Factor μP : RP → 1 : 33.3**
    - Cluster of 100 μPs = four 19-inch racks
      - » footprint = 6 square feet
    - Reconfigurable computer (SRC MAPstation™)
      - » footprint = 1 square feet

Savings of HPRC
(Based on one Cray-XD1 chassis)

Application	Speedup	SAVINGS		
		Cost Savings	Power Savings	Size Reduction
Smith-Waterman (DNA Sequencing)	2794	28x	140x	29x

- **Assumptions**
  - **100% cluster efficiency**
  - **Cost Factor µP : RP → 1 : 100**
  - **Power Factor µP : RP → 1 : 20**
    - Reconfigurable processor (based on one XD1 Chassis): 2200 W
    - µP board (with two µPs): 220 W
  - **Size Factor µP : RP → 1 : 95.8**
    - Cluster of 100 µPs = four 19-inch racks
      - » footprint = 6 square feet
    - Reconfigurable computer (one XD1 Chassis)
      - » footprint = 5.75 square feet

Outline

- Introduction
- Implementation Approach
- Testbeds
- Experimental Results
- **Conclusions**

Conclusions

- Potential of using multi-node HPRCs for computational biology applications investigated
- Scalability issues for S-W algorithm were characterized
- Orders of magnitude speedup demonstrated
- Scalability on both machines proved almost ideal when the number of nodes increased
- As number of nodes exceed a certain limit, scalability will decrease due to communications overhead
- FPGA local memory still relatively small compared to the problem size