BIPS

COMPUTATIONAL RESEARCH DIVISION

Performance Complexity: A New Execution Time Metric

Erich Strohmaier
Future Technology Group
EStrohmaier@lbl.gov

HPEC 2006, Sep. 21, 2006

Office of Science
U.S. DEPARTMENT OF ENERGY

Performance Complexity

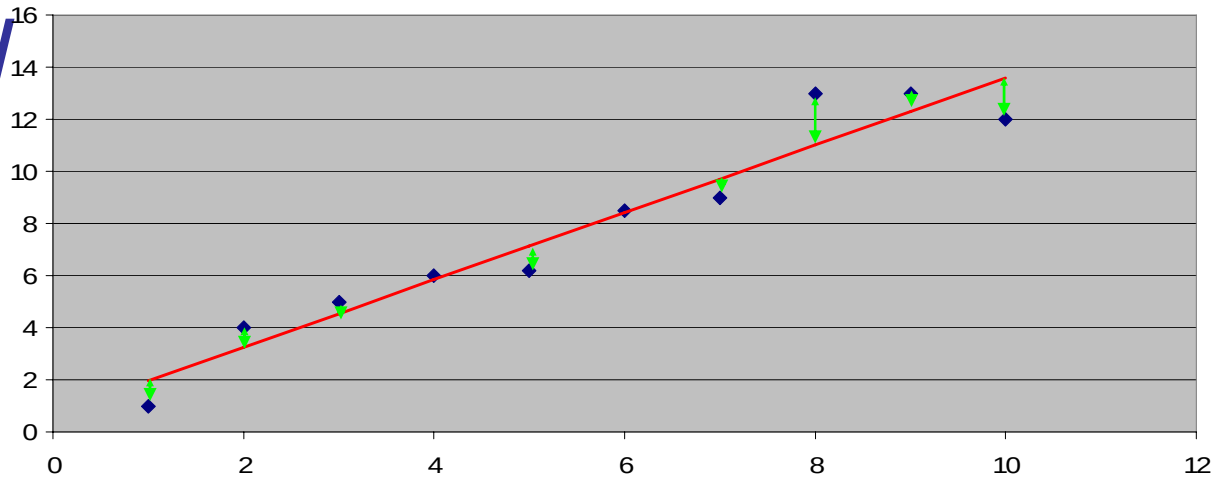
- The Programming Complexity of our systems is of growing concern.
  - At least with respect to performance.
  - Performance transparency is sinking!
- Can we develop a measurement based metric for PC?
  - The values in this metric will depend on the codes we use for measuring.
    - Just like performance does.

Performance Modeling

- **Performance Models** help us to learn about and show how well we understand:
  - algorithms,
  - Systems, and
  - programming environments.
- Can we use them to give us a measure for:
  - How complex our systems are?
  - How difficult they are to program (from a performance point of view)?

A Measure for PC

- If performance of a system is **predictable**, it is easy controllable and 'programmable'.
- **Sum of Squared Errors** (SSE) measures how well we model our performance data.
- Can we use SSE to compare performance transparency/ programming complexity?



A Measure for PC

- Use a defined "*benchmark set*".
- **Measure** performance of these benchmarks.
- **Predict** performance with *performance models*.
- Use a metric derive from SSE for the whole set to compare systems.
  - Do sound statistics (log-transformations).
- **Define two execution time metrics**:
  - **Performance** (P) and
  - **Performance Complexity** (PC) in same dimensions.

P and PC Metrics

$\overline{P} = \exp(\overline{\log(P)}) = \sqrt[n]{\prod_n P_j}$	**[ops/cycle]**
Absolute $PC_a = \exp(\sqrt{\overline{SSE(\log P, \log M)}}) - 1$	**[ops/cycle]**
Relative $PC_r = \exp\left(\sqrt{\overline{SSE'} / \overline{SS'}}\right) - 1$	**[]**

"Geometric mean" and "geometric standard deviation"

A Measure for PC

- As first exercise we use a synthetic parameterized benchmark - APEX-Map!
  - Each parameter set represents a benchmark.
  - Only one set of performance models.
  - Still spans a nice range of execution requirements.
- Our metric is "cycles/memory access".

Performance Evaluation

- Local and global memory access is the dominant performance bottleneck for many scientific applications.
- Synthetic benchmarks are suited to investigate details of data access as they are easier to handle and understand than application benchmarks.
- Dependency of performance on spatial and temporal data locality features is of great interest.
- Parameterized benchmarks allow to explore the dependency of performance of a whole range of descriptive parameters by generating multi-dimensional performance maps.

Application Performance Characterization (Apex)

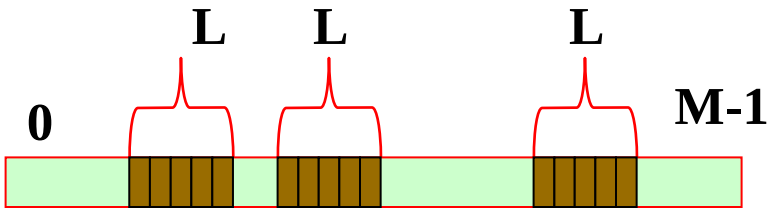
- Characterize a data access stream quantitatively with as few parameters for spatial and temporal locality as possible.
  - Use the same concepts for temporal and process locality for parallel execution.
  - Simulate temporal locality by more frequent accesses to certain memory locations.
  - Ignore other effects such as sharing of variables or process affinity.
- Create a parameterized synthetic benchmark, which generates a synthetic stream of references, to investigate data access performance.

Locality Concepts

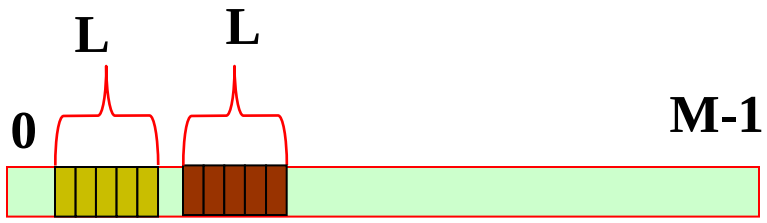
- Data set size: *M*
- Spatial Locality (*L*):
  - Blocked access to *L* contiguous data elements.
  - *L* is also the innermost loop length!
- Temporal Locality (α):
  - Achieve more frequent access to certain memory locations by using non-uniform random starting addresses of blocks distributed according to a power law.
  - Characterize temporal locality with the exponent α of the power law.

Memory Access Probe Apex-MAP

- **Use the Power distribution as non-uniform random address generator.**
  - **Self-similar and thus scale invariant.**
  - **Single descriptive parameter.**
  - **Exponent α in [0,1]**
    - **α=1 : Uniform random access.**
    - **α=0 : Access to a single vector only.**

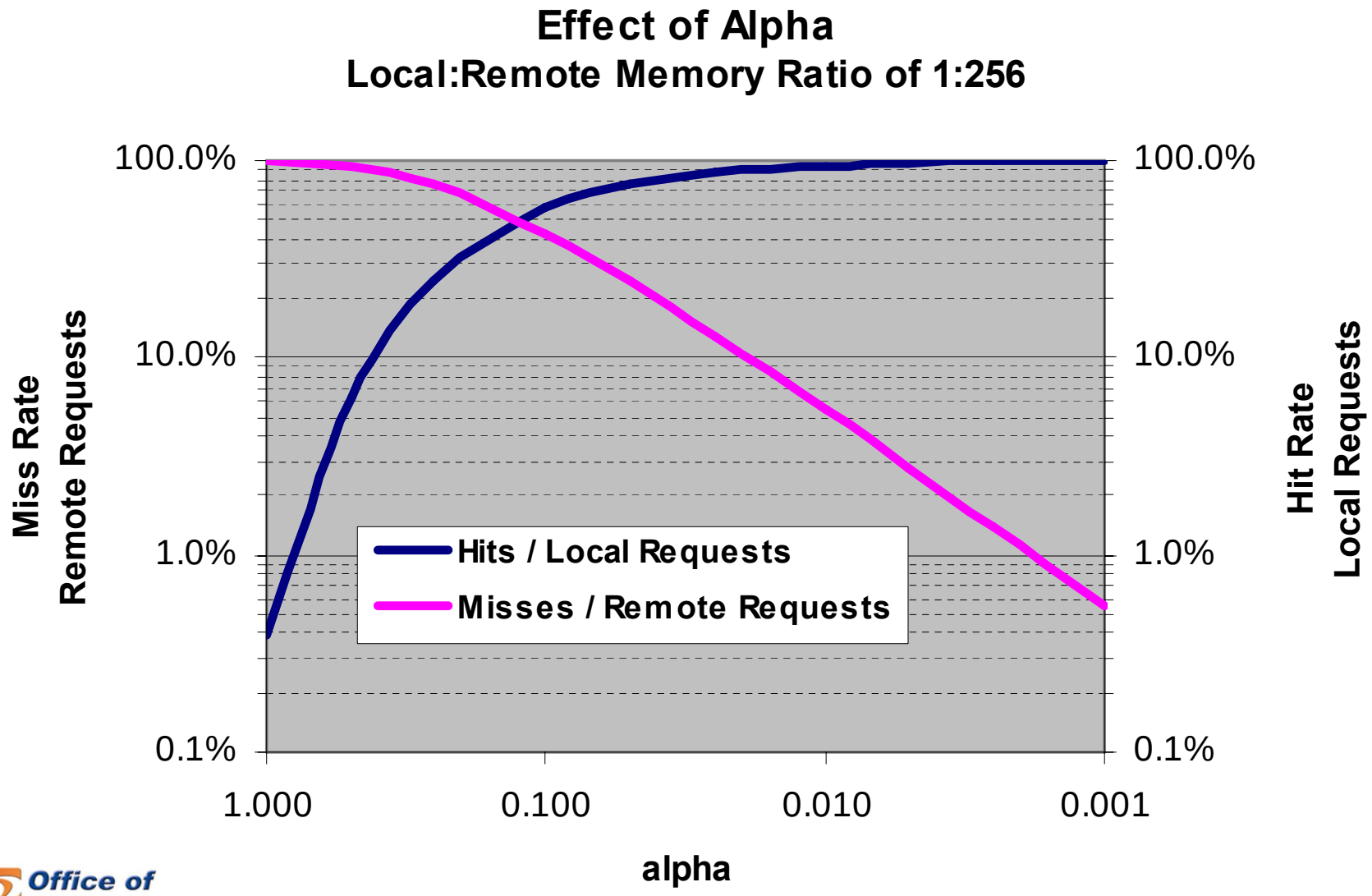


$\alpha \approx 1$



$\alpha \approx 0$

Effect of Alpha



APEX-Map Outline

Sequential:

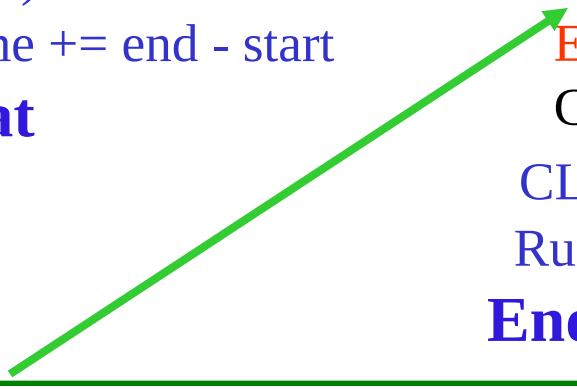
Repeat N Times
- **Generate Index Array()**
- **CLOCK(start)**
- **For each Index *i* in the Array**
  - Compute()
- CLOCK(end)
- RunningTime += end - start

End Repeat

Parallel:

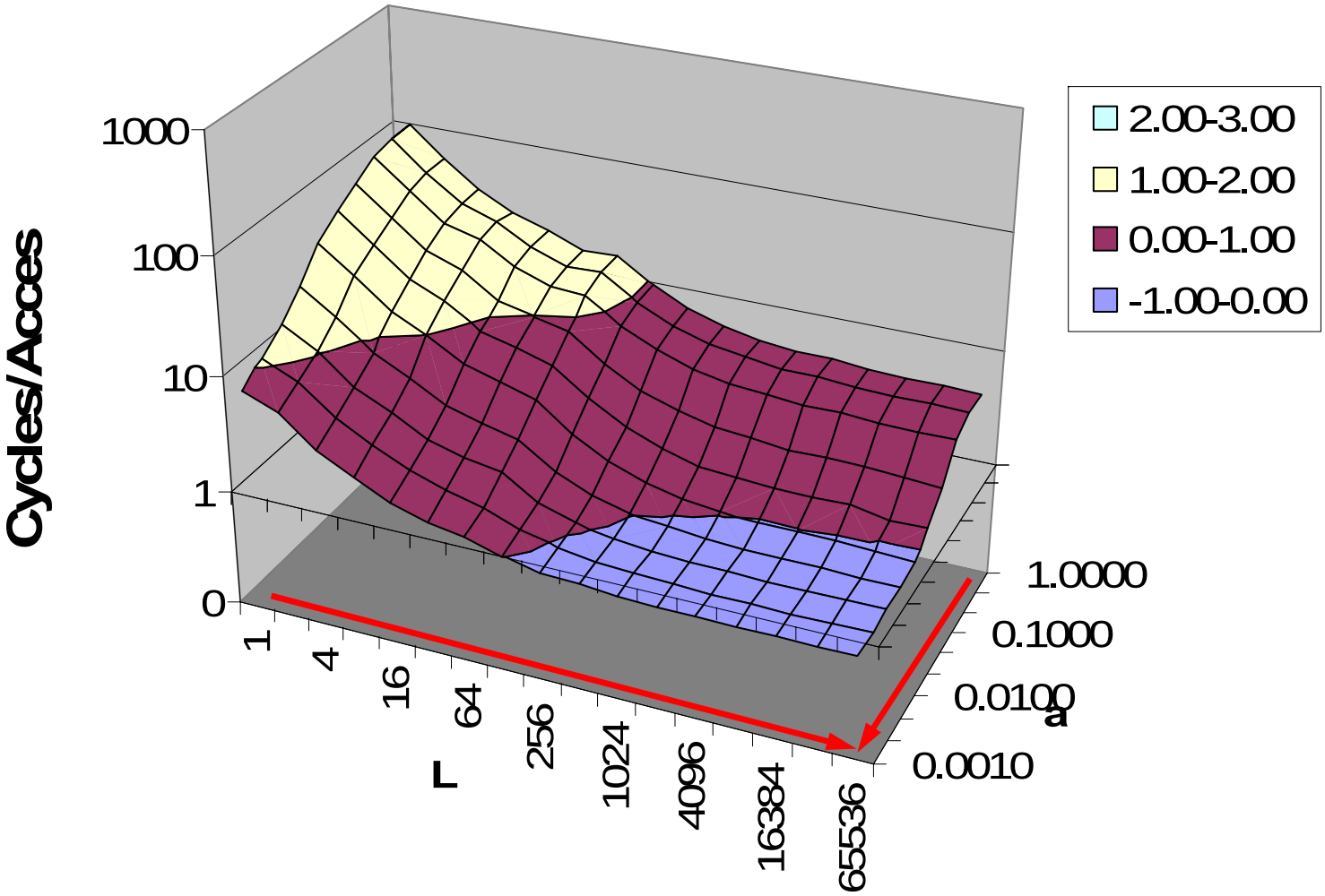
Repeat N Times
- **Generate Index Array()**
- **CLOCK(start)**
- **For each Index *i* in the Array**
  - If (remote data)
    - **GetRemoteData()**
  - End If
  - Compute()
- CLOCK(end)
- RunningTime += end - start

End Repeat

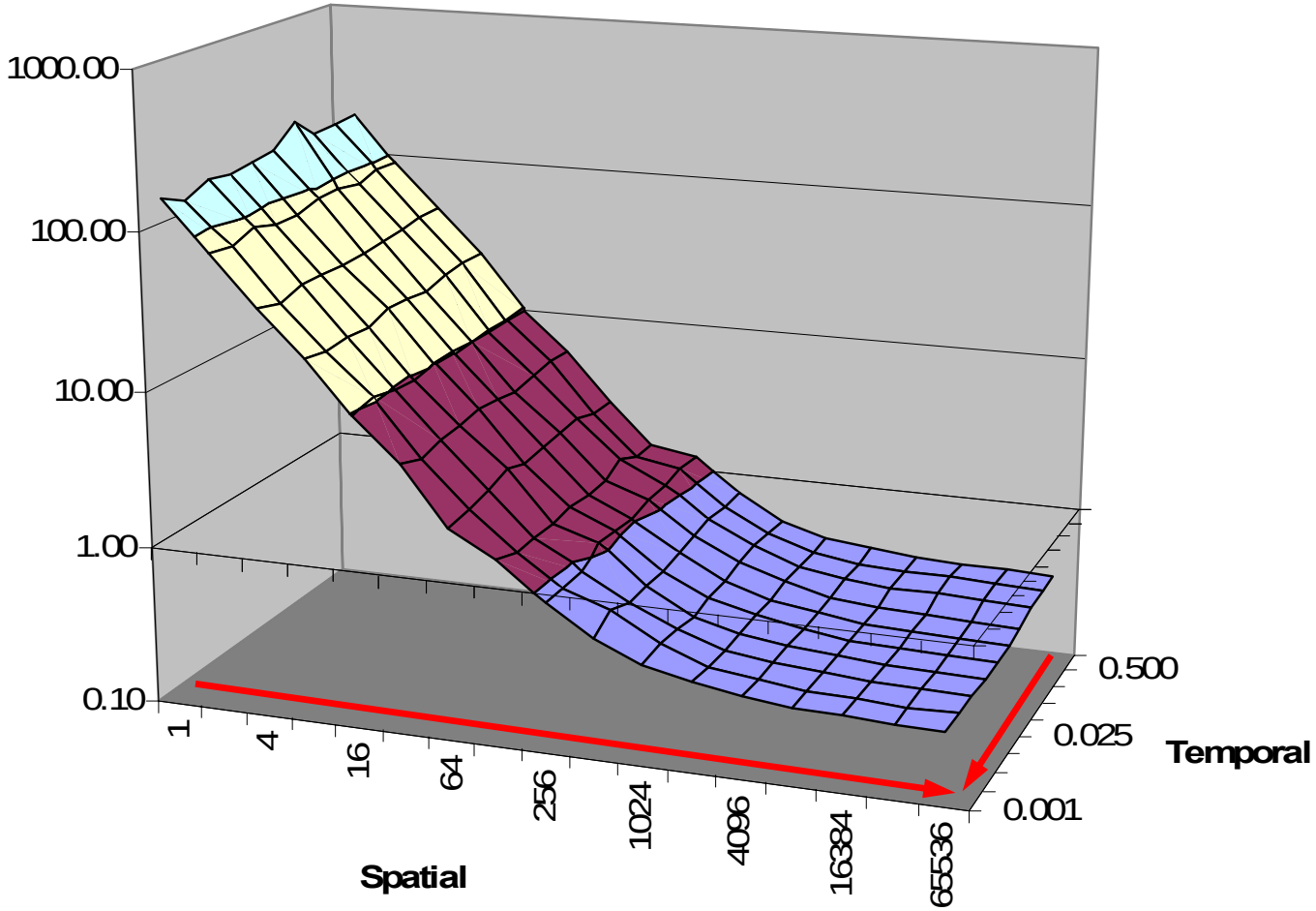


Depends on the chosen parallel programming model

Itanium2 Processor Profile



Cray X1 Processor Profile

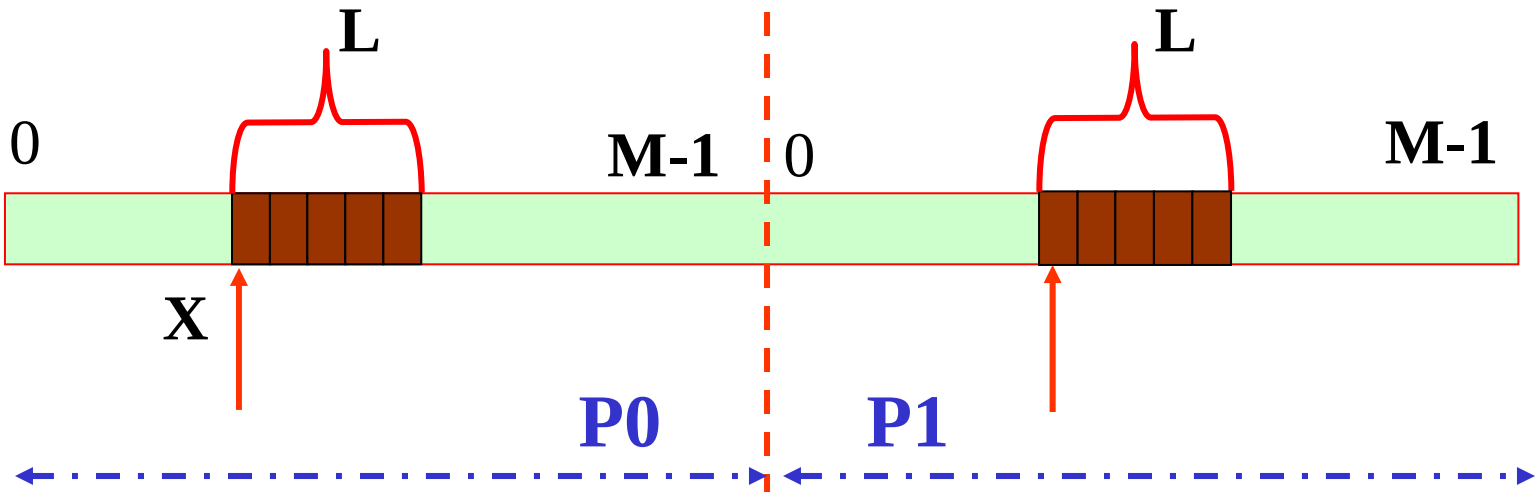


BIPS

Berkeley Lab

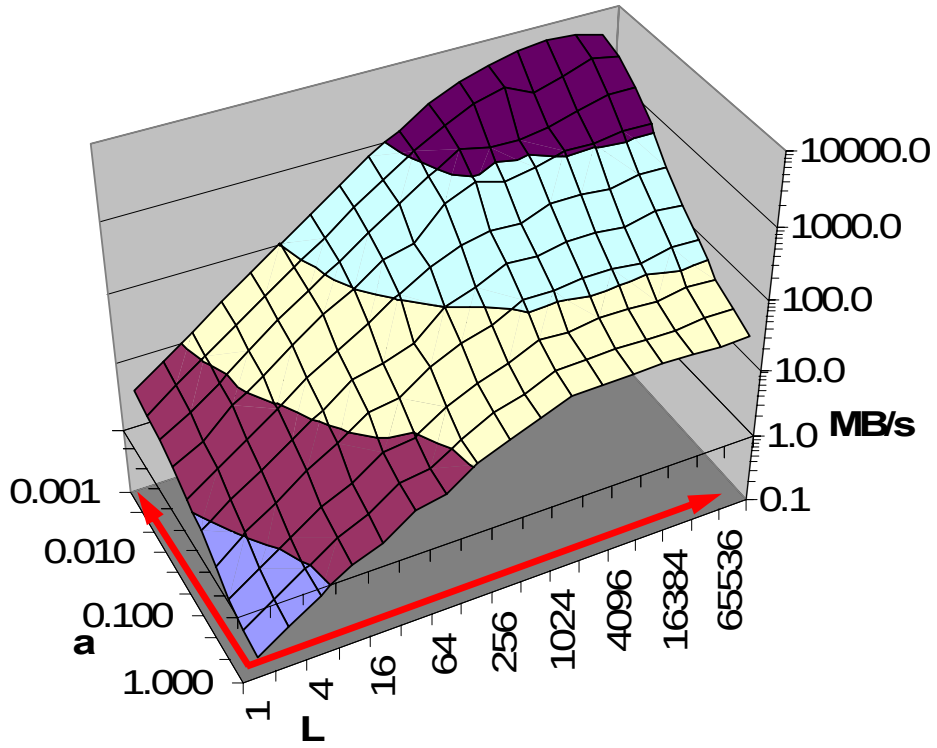
Office of Science
U.S. Department of Energy

Parallel APEX-Map Design

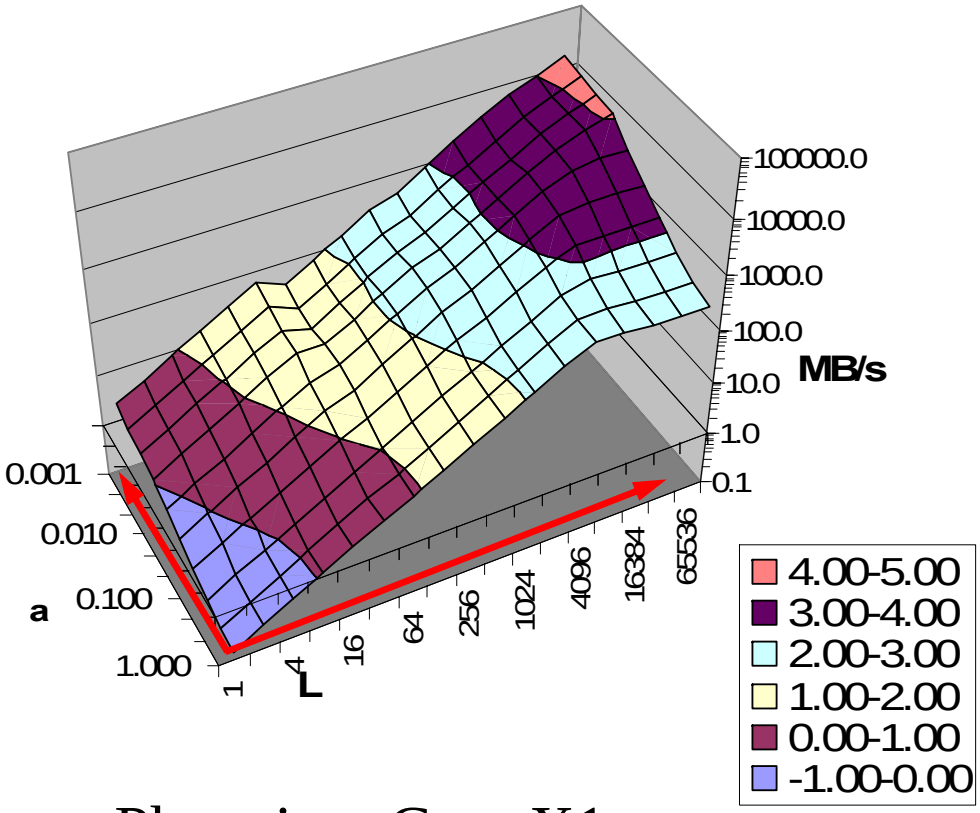


- Same parameters as sequential version (*α, L, M*P*).
- Data evenly distributed among processes.
- X distributed across all processes.
- Each remote access results in a message with length *L* (for the message passing paradigm).

Parallel Performance Surfaces
256 Processors - MPI



Cheetah – IBM SP Power4



Phoenix – Cray X1

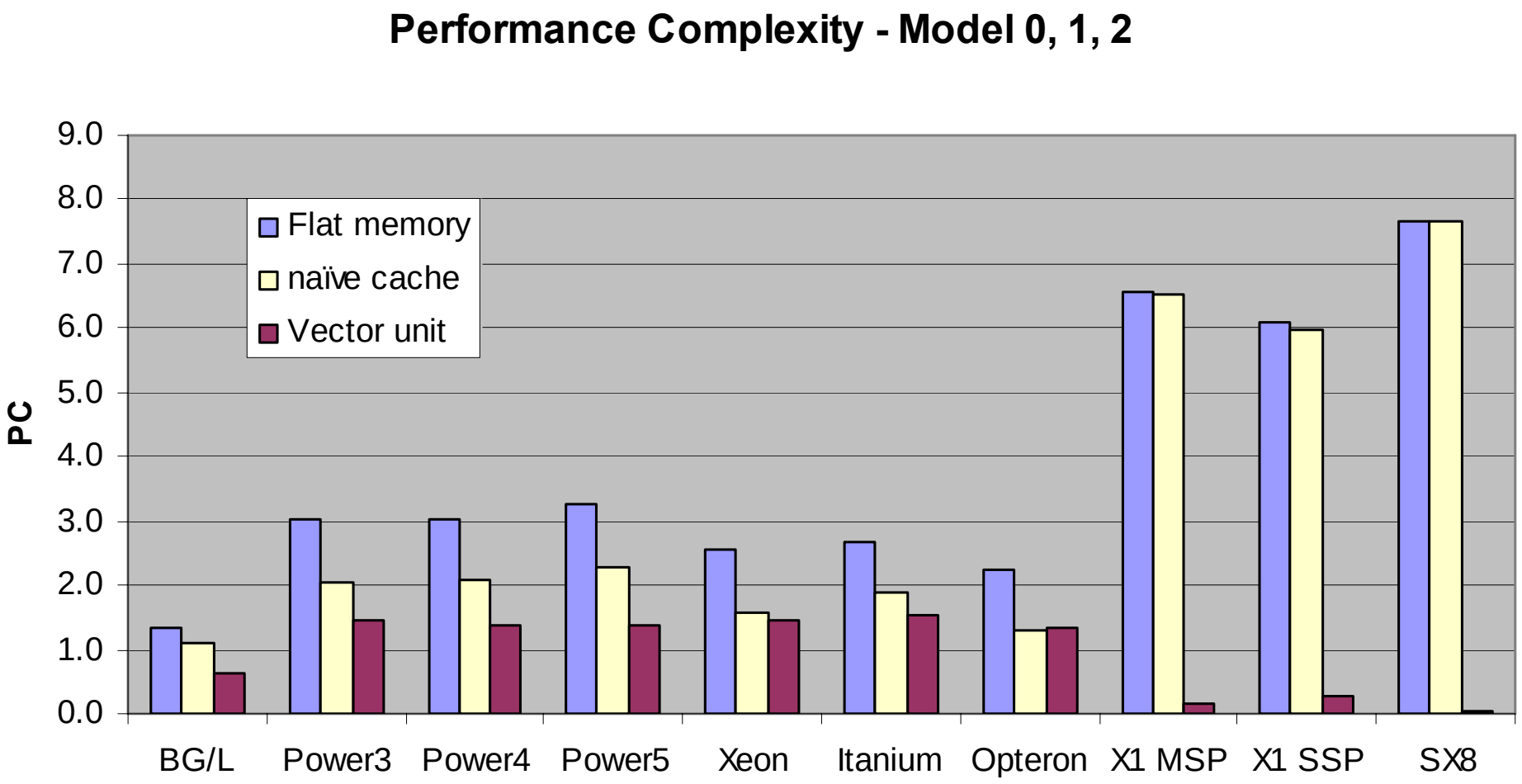
Limitations

- Biases of this study:
  - Benchmark selection (Only one code)
  - Parameter range selection
    - L and α
    - M = 512 MB/process fixed
    - I=1024
  - Power of 2 L parameter scaling
  - Model selection

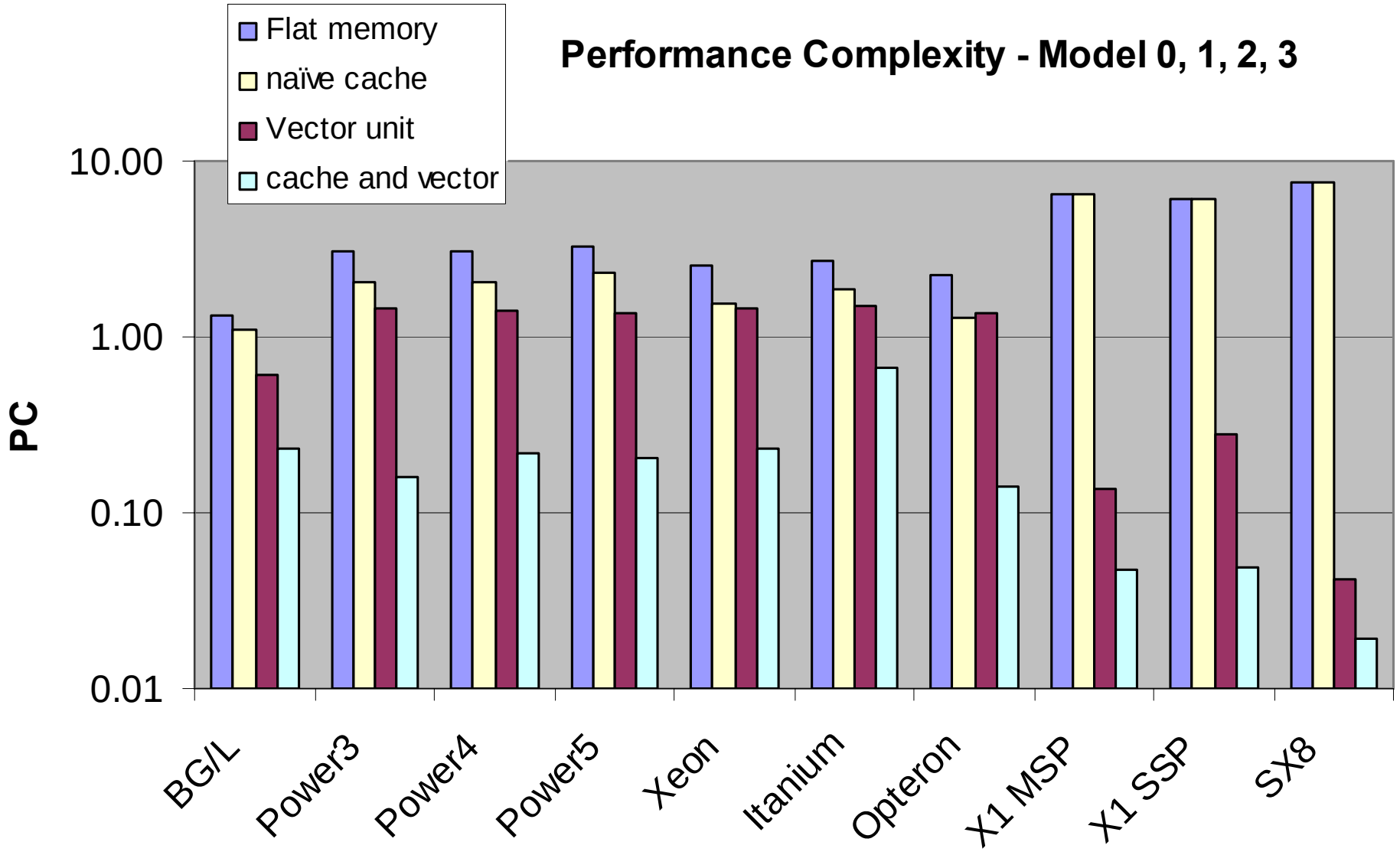
PC – 4 Models

- Use four performance models:
  - Ideal, flat memory
    - *T = const*
  - Two level memory hierarchy (c and m)
    - *T = P(c/m)*a + (1-P(c/m))*b*
  - Linear access timing to single level memory
    - *T = [a +(L-1)*b]/L*
  - Combined: Linear timing for two levels
    - *T = [P(c/m)*(a+b*(L-1)) + (1-P(c/m))*(c+d*(L-1))]/L*
- We back-fit effective values for parameters.

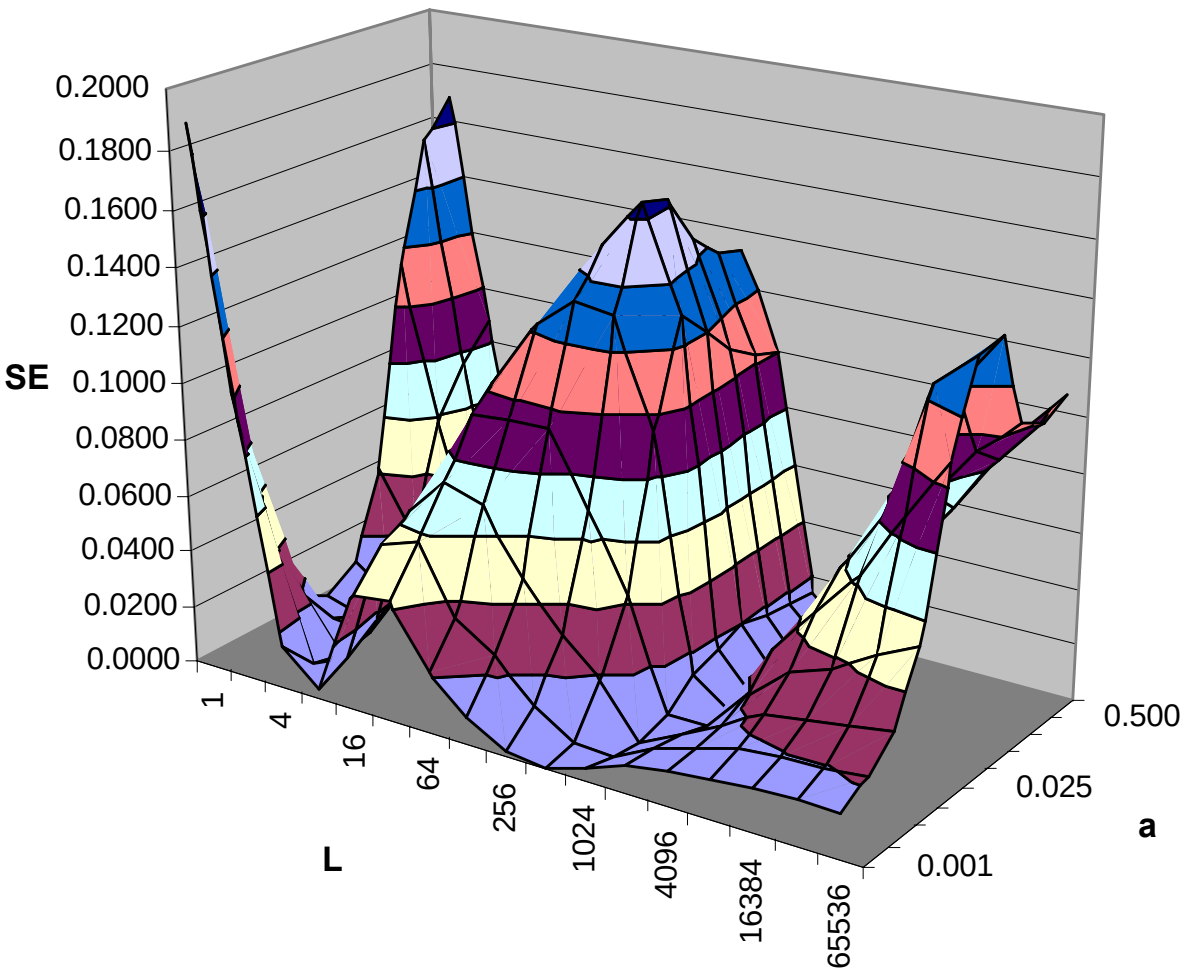
PC Values for 3 simple Models



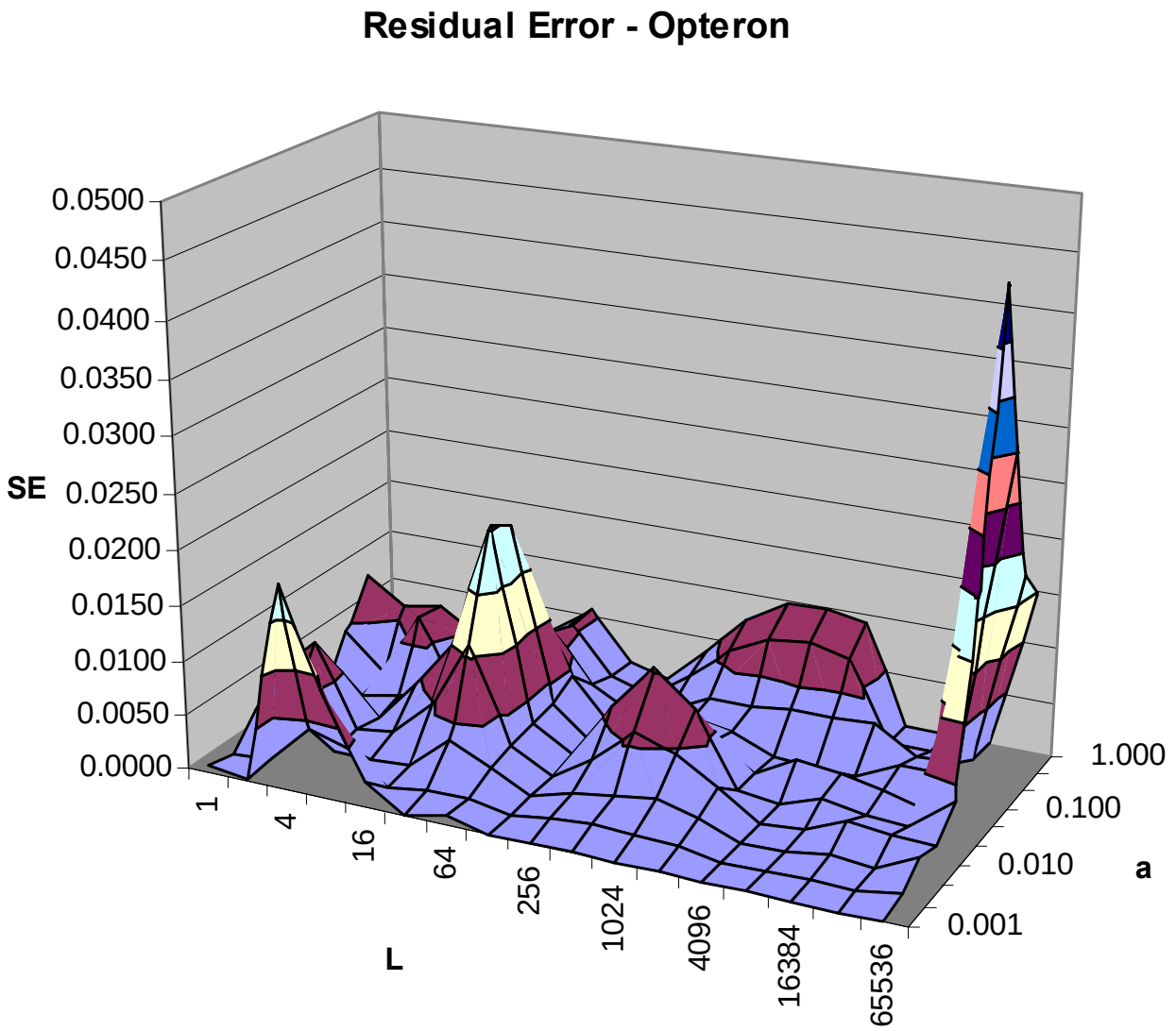
PC Values for 4 simple Models



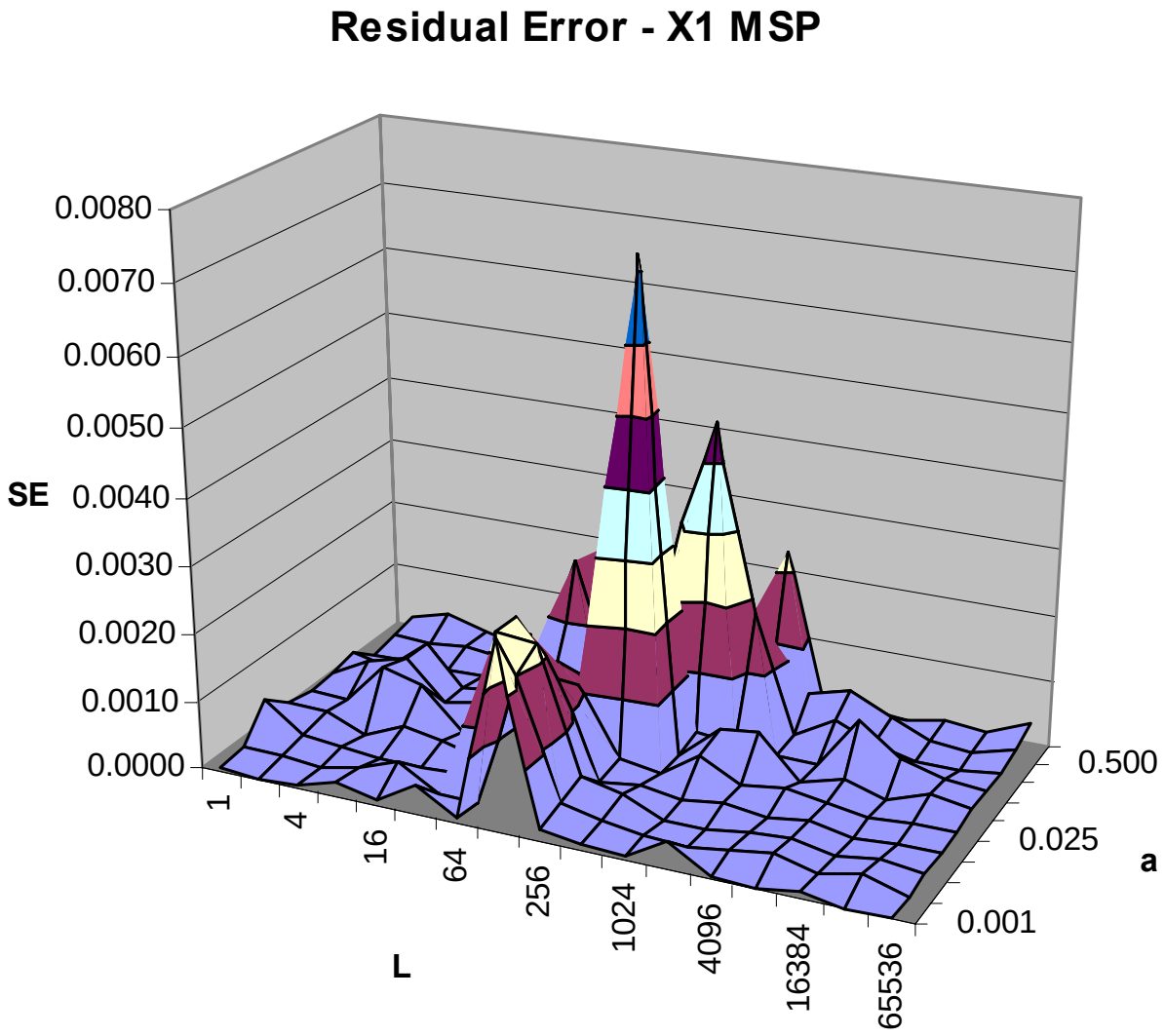
Residual Error - Sequential



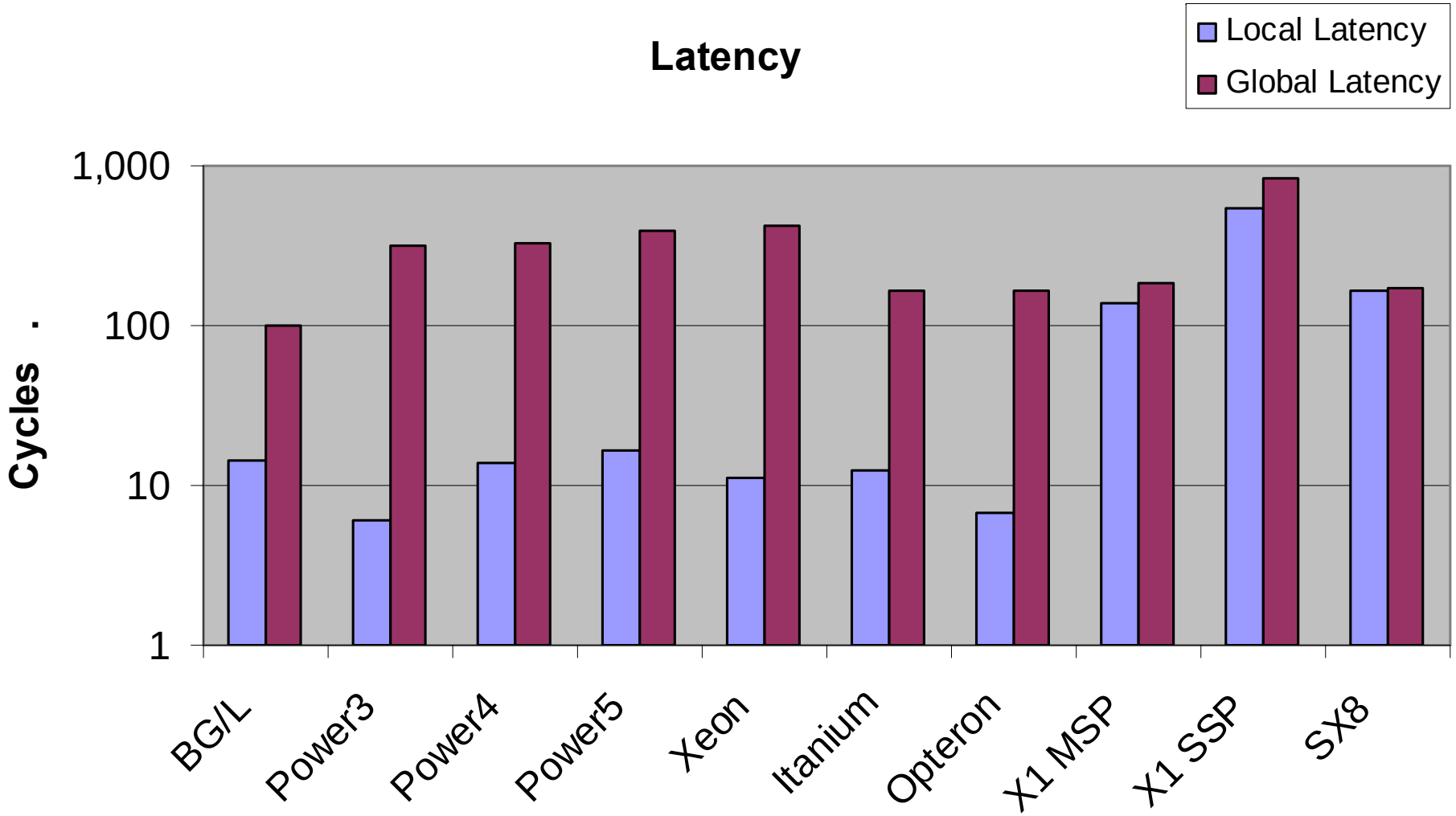
Residual Error - Sequential



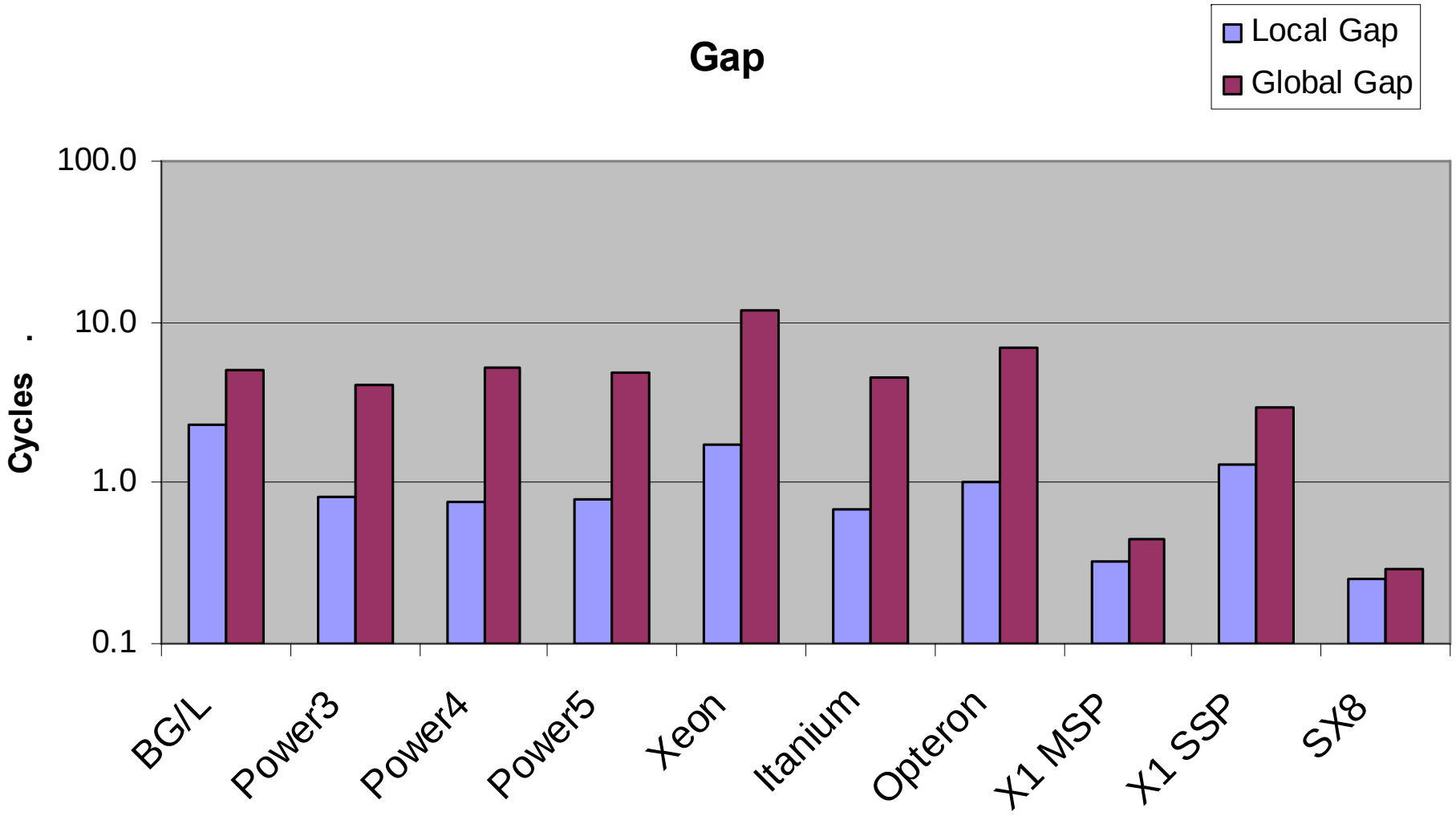
Residual Error - Sequential



Serial Perf. Model Parameters

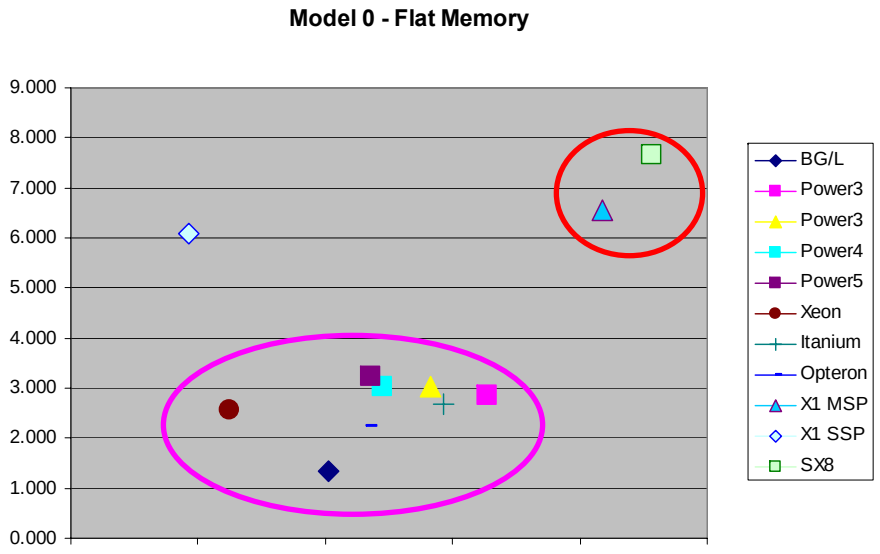


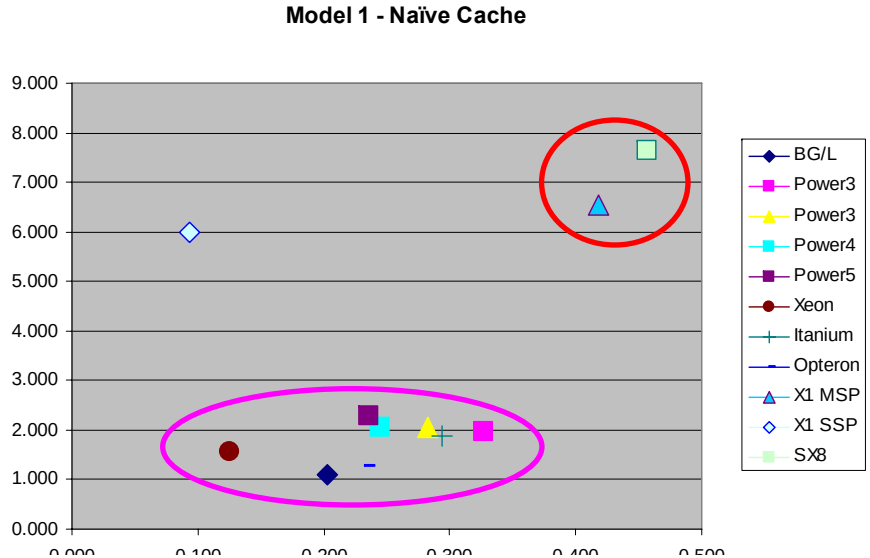
Serial Perf. Model Parameters

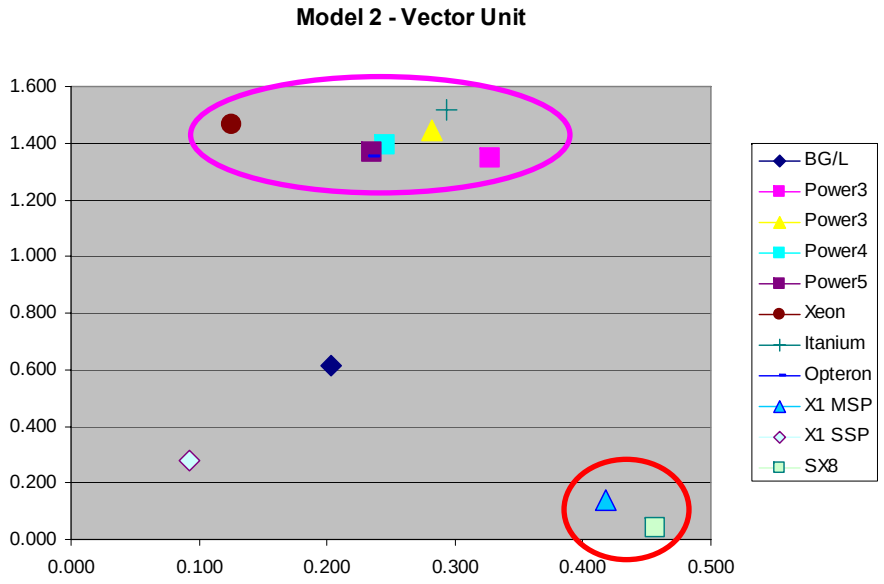


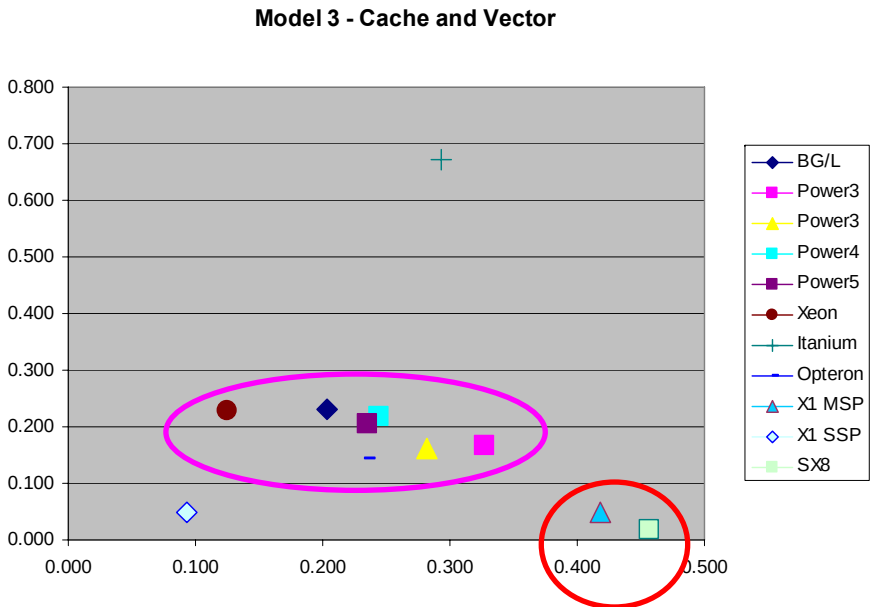
Lower is better

P-C Maps for all Four Performance Models for Serial Execution

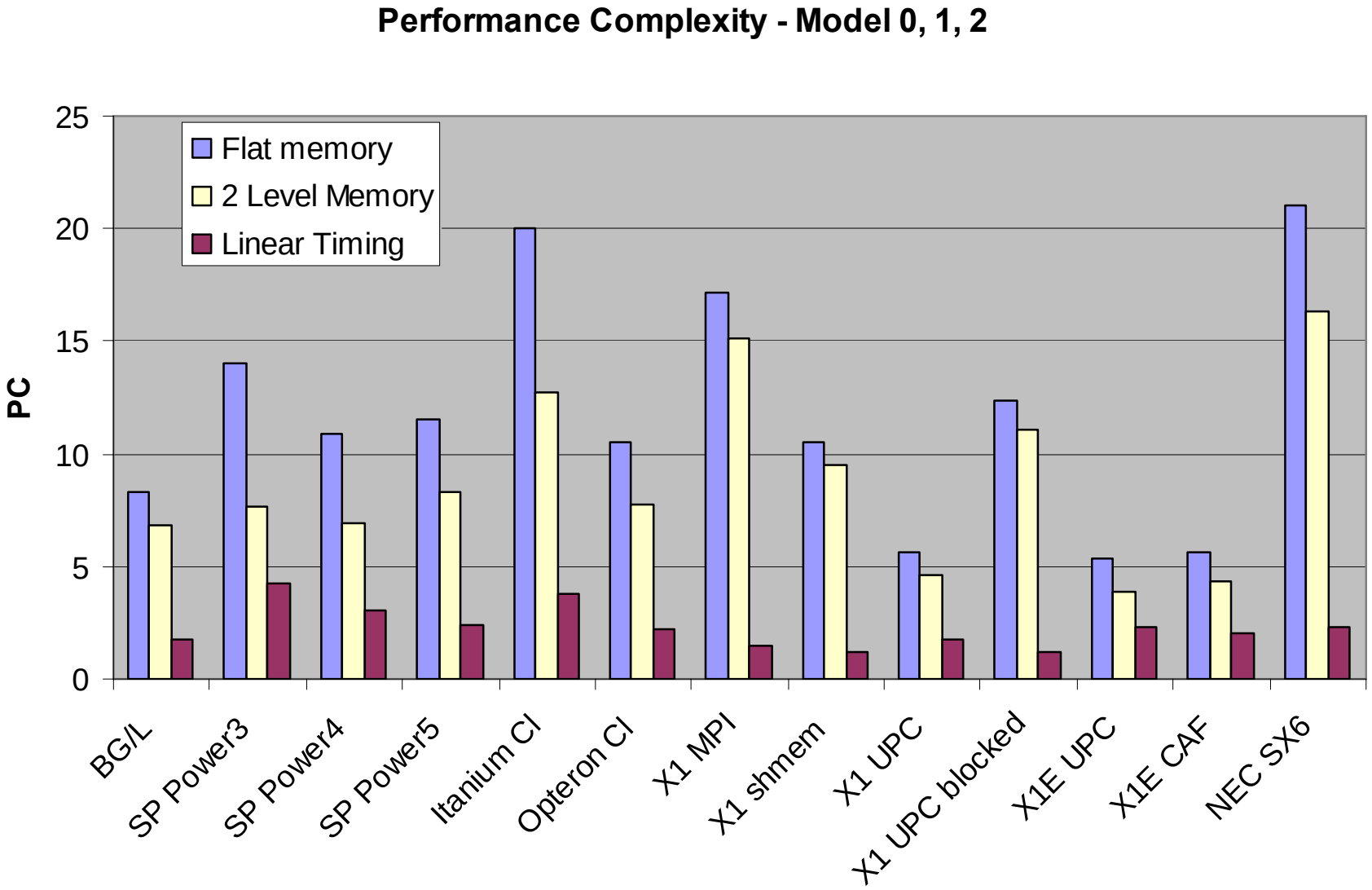




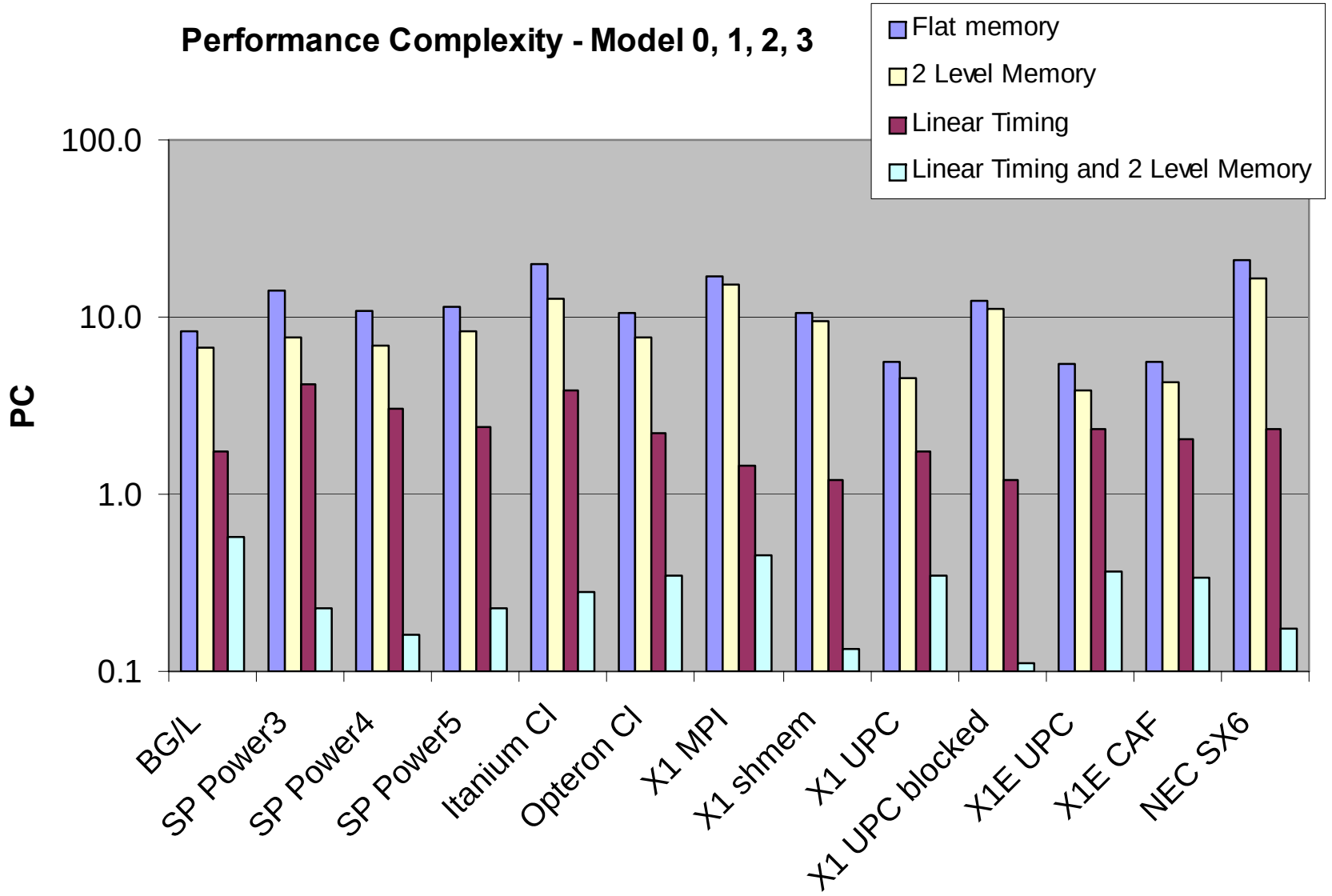




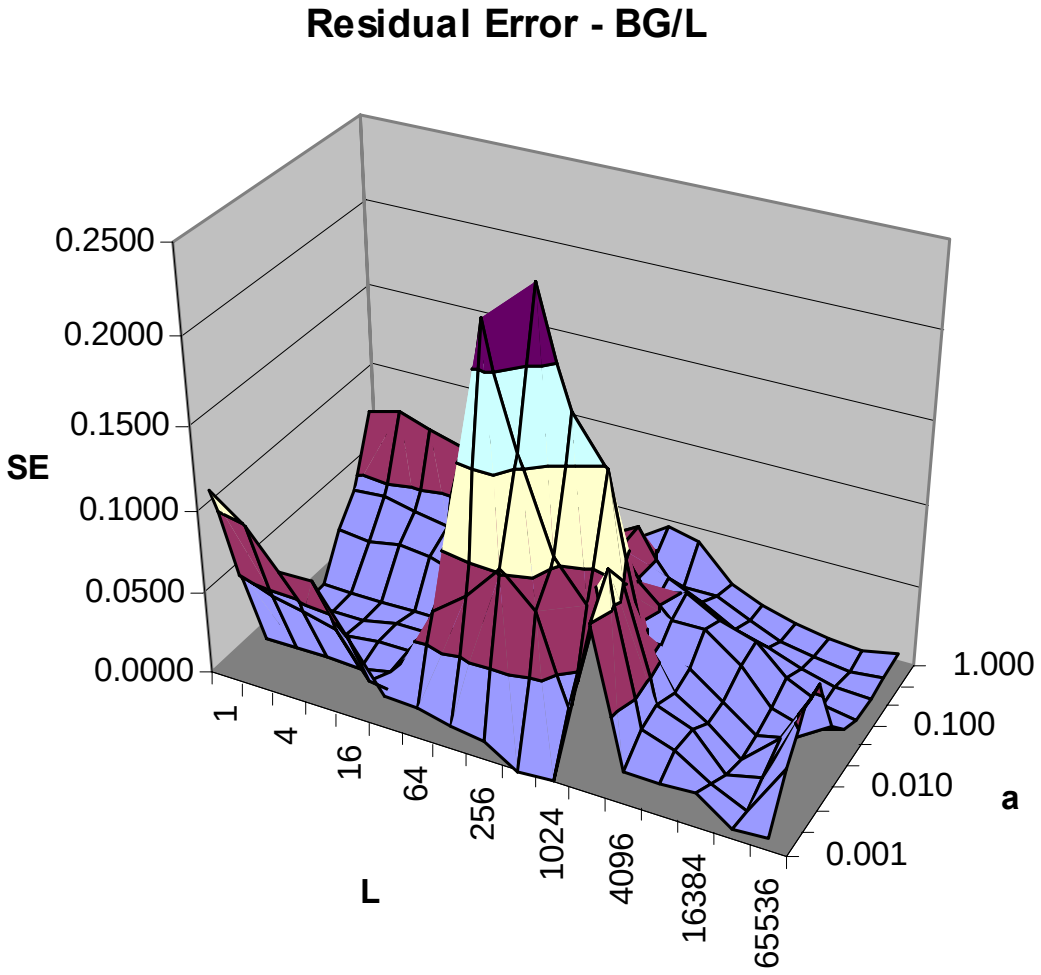
Parallel PC


Performance Complexity - Model 0, 1, 2
PC
25
20
15
10
5
0
Flat memory
2 Level Memory
Linear Timing
BG/L
SP Power3
SP Power4
SP Power5
Itanium CI
Opteron CI
X1 MPI
X1 shmem
X1 UPC
X1 UPC blocked
X1E UPC
X1E CAF
NEC SX6

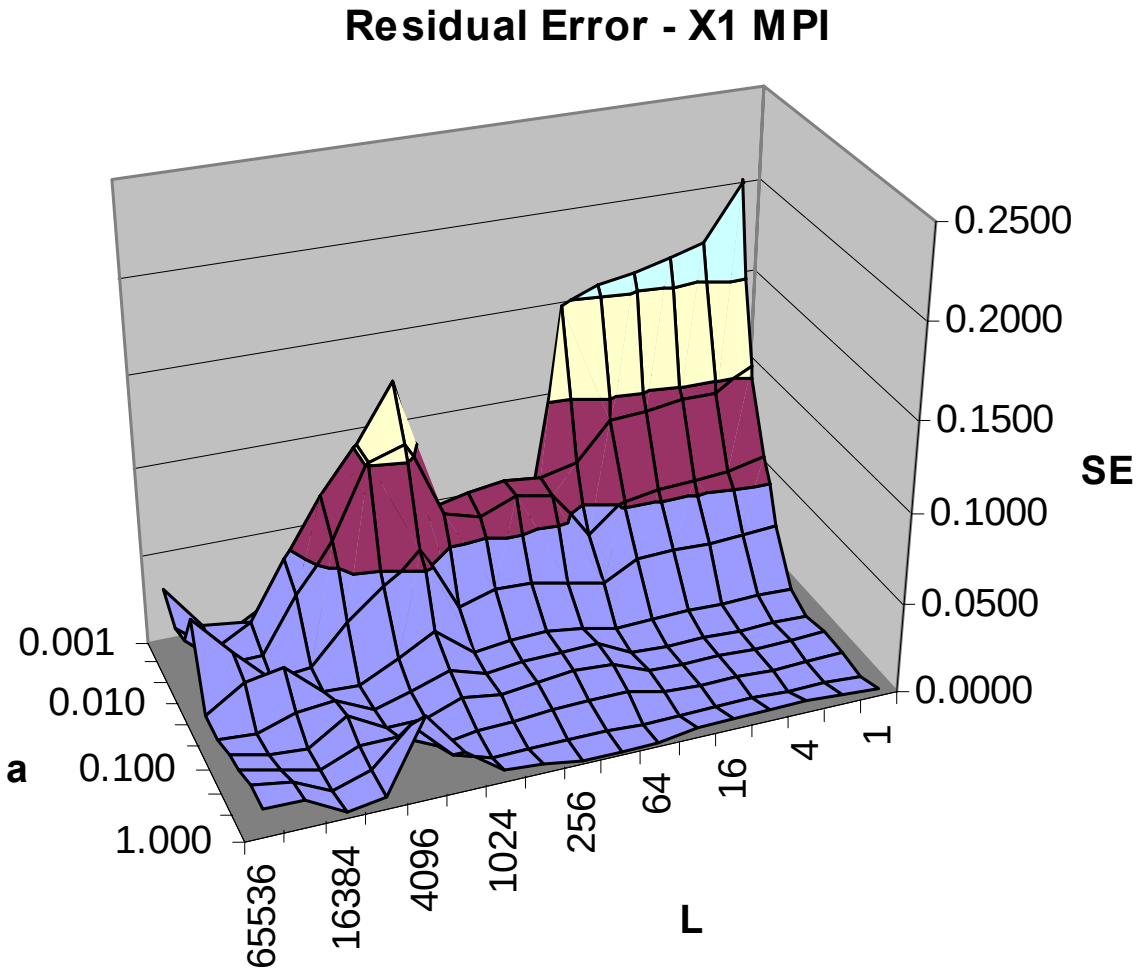
PC Values for 4 simple Models



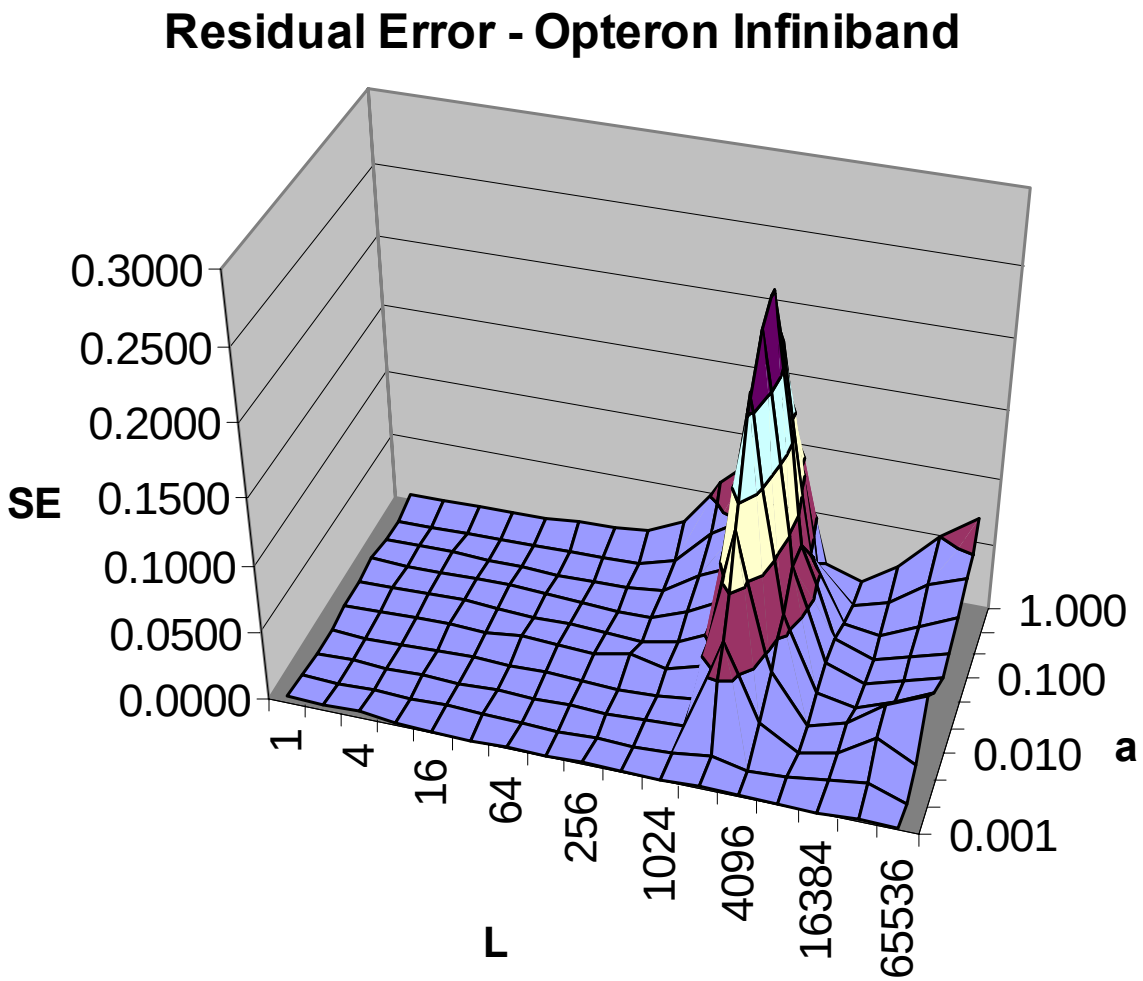
Residual Error - Parallel



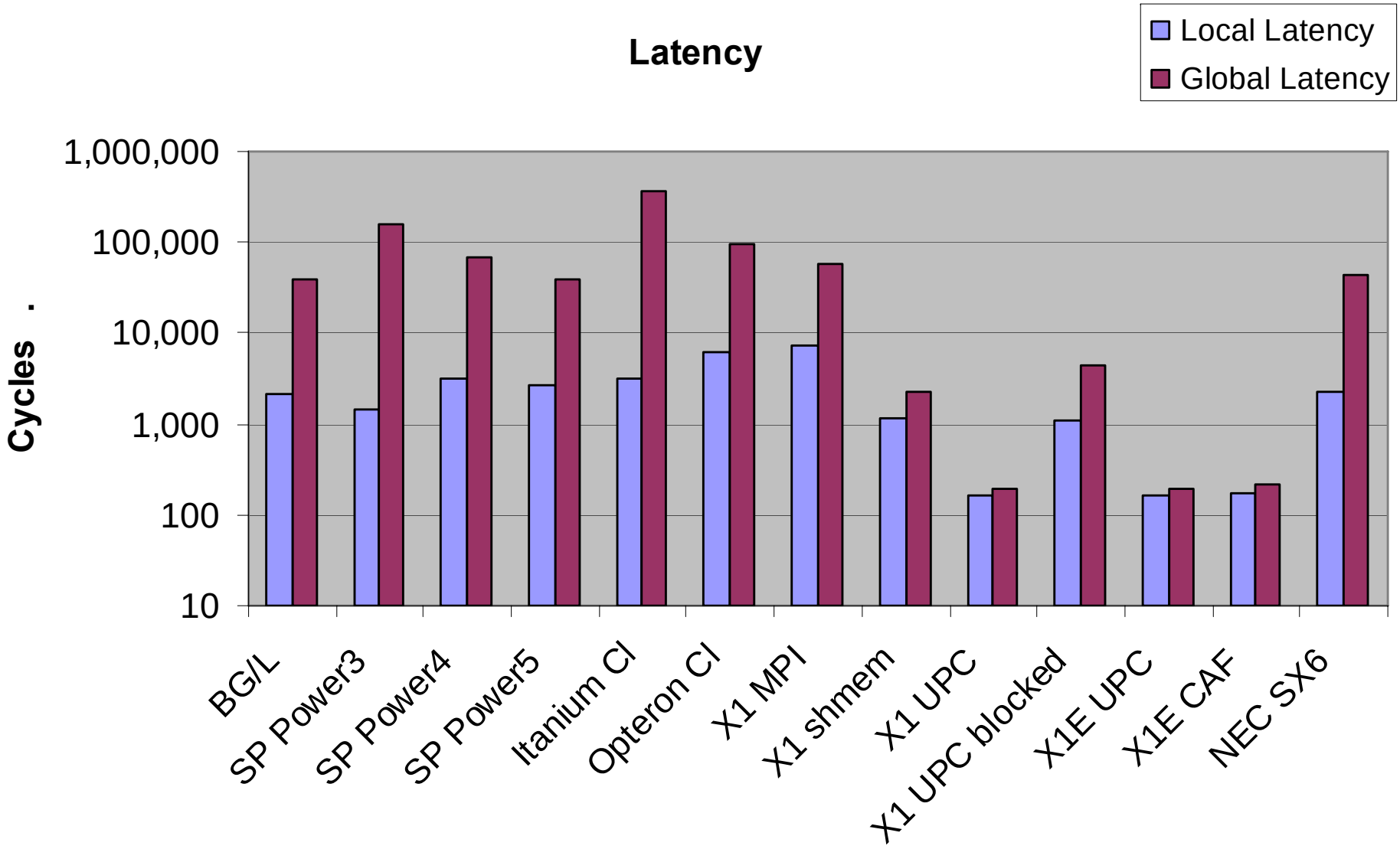
Residual Error - Parallel



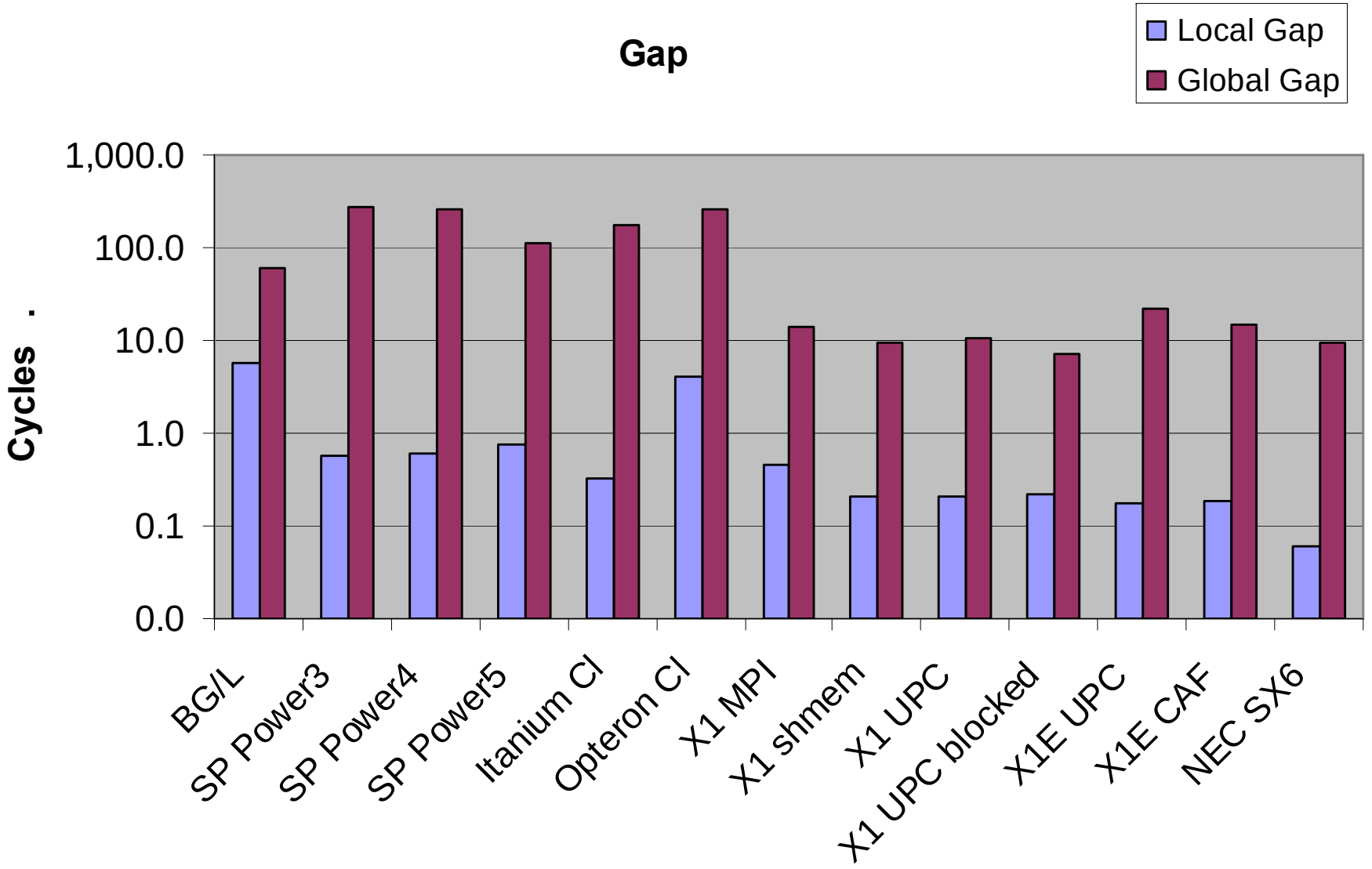
Residual Error - Parallel



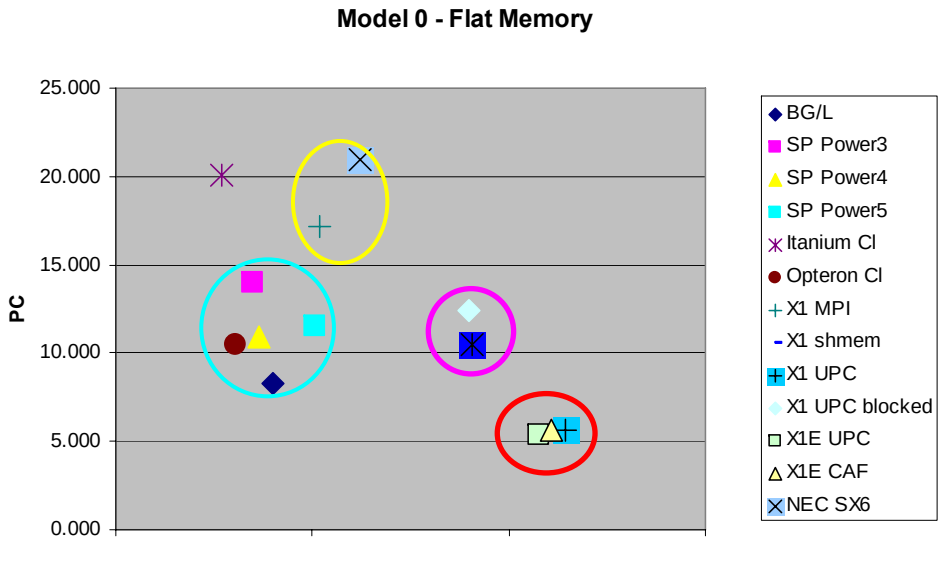
Parallel Perf. Model Parameter

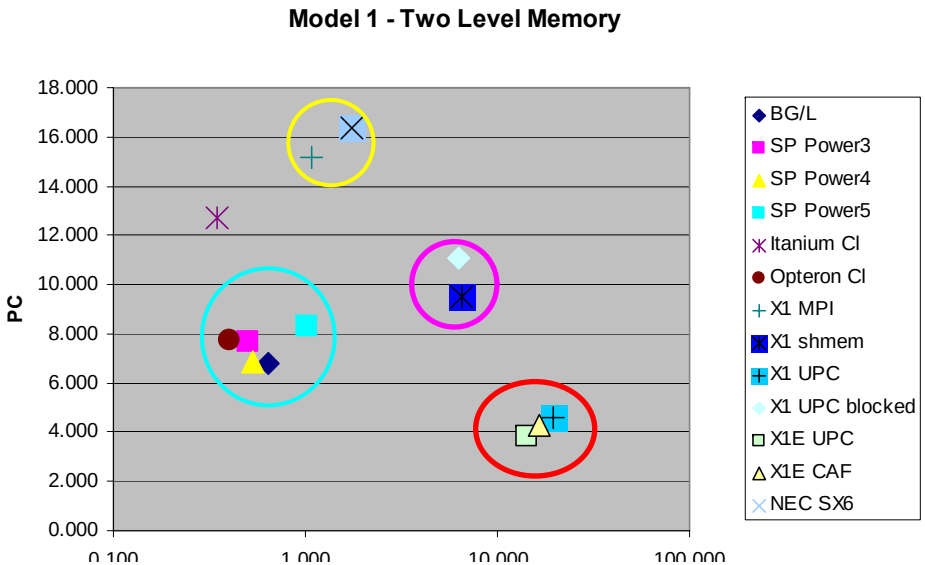


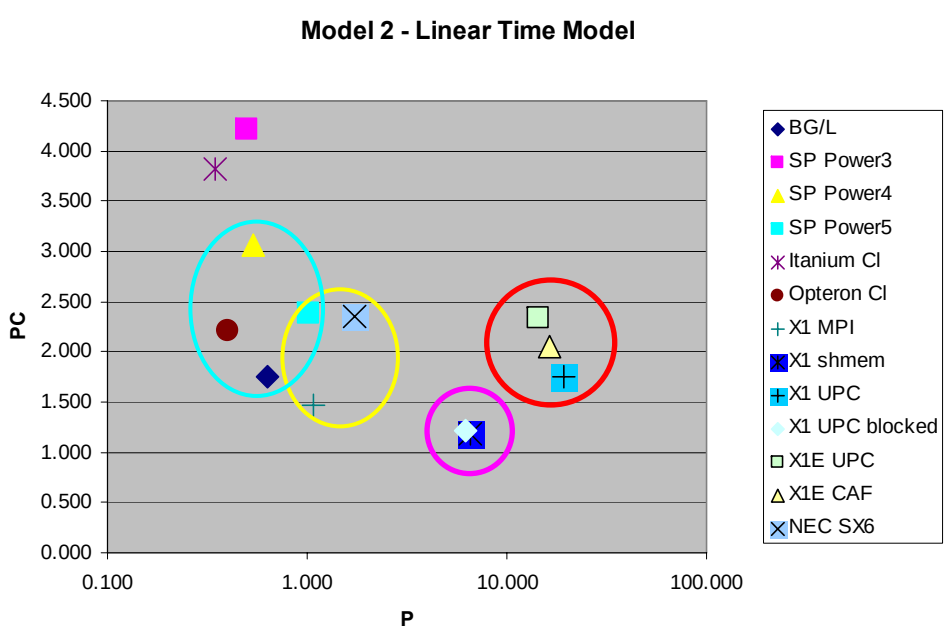
Parallel Perf. Model Parameter

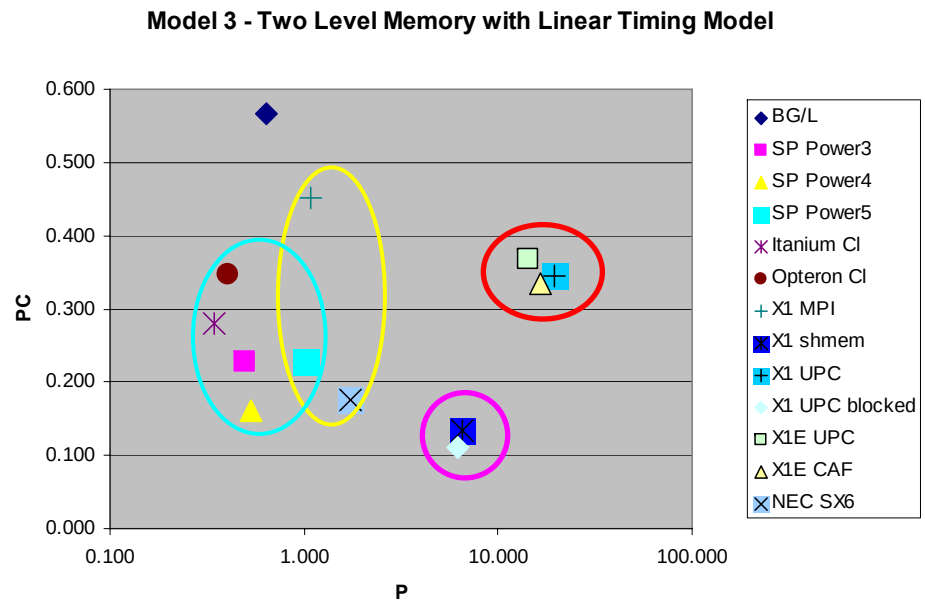


P-C Maps for all Four Performance Models for Parallel Execution









Conclusions

- The combination of benchmarks and performance models allows to characterize the performance complexity of systems quantitatively.
- Two dimensional performance complexity maps enable a high-level comparison of different systems.
- Performance models allow to analyze system behavior through residual errors in great detail.
- APEX-Map is a parameterized synthetic data access benchmark, which allows to map performance for a large range of temporal and spatial localities systematically.
- These performance maps reflect the performance impact of different system and memory features.