FORUM ON PARALLEL COMPUTING CURRICULA
Wellesley College, Massachusetts

Parallel Computing
Models, Languages, and Architectures

Azer Bestavros

Computer Science Department

Boston University

Friday March 31^{th}, 1995

† This work is supported partially by the NSF.

Motives for Developing the Course

The emergence of different parallel computer architectures has led to the adoption of various programming models and languages. These different models pose a number of problems for parallel program design:

- ◇ It is unclear whether the specification of parallel algorithms transcends the differences between the architectures on which they will be implemented and the application domains from which problems are drawn.
- ◇ It is unclear whether the underlying architectural model needs to be reflected through to the programmer, or whether the programmer can write a machine independent program that would perform well on different architectures.
- ◇ It is unclear what kinds of linguistic constructs should be provided to the programmer to help in constructing efficient yet understandable programs that can run either on a particular architecture or on parallel machines in general.

The Course

- **Goals to be achieved**:
  - Give students a *balanced exposure* to advances in parallel computing.
  - Highlight the interplay between parallel architectures and Parallel programming models and languages.
  - Contrast *Covert* and *Overt* parallelism and introduce parallelizing compiler techniques.
  - Hands-on experience to elucidate some of the above points.

- **Hurdles**:
  - There is not a single textbook that satisfies the above needs.
  - Many of the programming exercises have to be on simulators (if possible) or with pencil and paper.

- **Adopted Approach**:
  - Adopt a textbook that serves as a survey to supplement the lectures, which would provide the needed depth and insight.
  - As a course project, ask students to choose from a set of public-domain simulators of various systems and experiment/report on their experience by the end of the course.

Excerpts from the Course Syllabus

◇ **Objective:**
This course examines a palette of parallel programming models and languages that are suitable for different architectures and applications. Architecture independent programming methodologies will be contrasted to those targeted at Data-flow, Data-parallel, Pipelined, Shared Memory, or Message Passing architectures. Lab assignment provide hands-on experience with a host of parallel programming languages.

◇ **Prerequisites:**
CS-420: Introduction to parallel computing is required (could be waived).
CS-320/CS-520: An exposure to Programming Languages is recommended.
CS-450/CS-550: An exposure to Computer Architecture is recommended.

◇ **Topics:**
The material covered encompassed topics in parallel computer architectures, parallel programming models, and languages. Appropriate examples for existing or proposed systems were surveyed and compared.

◇ **Textbook:**
Advanced Computer Architecture: Parallelism, Scalability, Programmability Kai Hwang, McGraw-Hill Series in Computing Science, 1993.

Schedule

Class	Topic	Readings
9/13	First class meeting: Motivation and Overview	Ch.1
9/15	PRAM and VLSI complexity models	Ch.1
9/20	Software Parallelism: Data and Control Flow	Ch.2
9/22	Hardware Parallelism: Network Architectures	Ch.2
9/27	Scalability: Applications and Metrics	Ch.3
9/29	Scalability: Speedup laws and Analysis	Ch.3
10/04	Processors and Memories: Basic and Superscalar Processors	Ch.4
10/06	Processors and Memories: Memory Management and Hierarchy	Ch.4
10/11	Multiprocessing: Communication and Memory Design Issues	Ch.7
10/13	Multicomputing: Overview and Design Issues	Ch.7
10/18	Multivector: Techniques and Examples (Cray & Futjitsu)	Ch.8
10/20	SIMD Processing: Techniques and Examples (CM-2 and MasPar)	Ch.8
10/25	MIMD Processing: Techniques and Examples (CM-5)	Ch.8
10/27	Shared Memory Processing and Multithreading	Ch.9
11/01	Fine-Grain Multicomputers: MIT J-M/C and Caltech Mosaic C	Ch.9
11/03	Multithreaded Scalable Machines: Dash, KSR-1, and Tera	Ch.9
11/08	Dataflow and Hybrid Machines	Ch.9
11/10	Parallel Programming Models: An overview	Ch.10
11/15	Architecture Independent Models: Unity	Handout
11/17	Parallelizing Compilers: Dependence Analysis and Scheduling	Ch.10
11/22	Parallelizing Compilers: Loop Transformations and Pipelining	Ch.10
11/24	Fall Recess (Thanksgiving)	
11/29	Programming: Shared Memory and Message Passing Paradigms	Ch.11
12/01	Programming: Decomposition Techniques	Ch.11
12/06	Operating Systems and Environments: An Overview	Ch.12
12/08	(Slack)	
12/13	(Slack)	
12/19	Final exam	

Sample Exercise from a Problem Set

<u>Problem *i*</u> (MIMD)

One technique to print all the prime numbers smaller than a given bound, M, can be sketched as follows:

1. Generate a ***big enough*** list of integers.
2. Eliminate any integer that is divisible by another with a smaller index.
3. When no more elimination is possible, print out uneliminated integers $< M$.

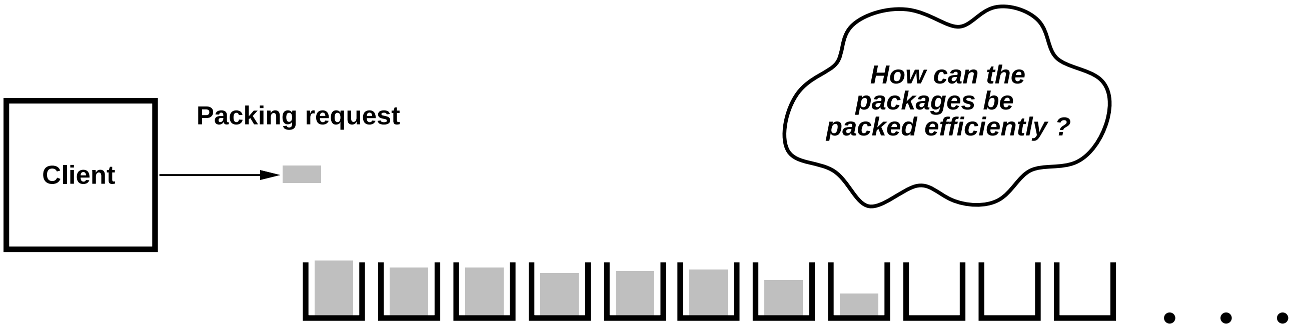
One way of eliminating non-prime numbers is to use a generator to spawn processes that would do the elimination.

- ◇ Write an *efficient* **BBN Butterfly** program that prints all prime numbers less than some given upper bound, M, using the aforementioned technique.
- ◇ Write an *efficient* **CM-5** program that prints all prime numbers less than some given upper bound, M, using the aforementioned technique.
- ◇ Which of the above two architectures would you choose to implement this algorithm? Which one do you think will be more efficient?

Hint: Use a generator to spawn processes that would do the elimination.

Sample Exercise from a Lab Assignment: Bin Packing

Parallel Simulation of Bin Packing on the CM



Bins of constant capacity

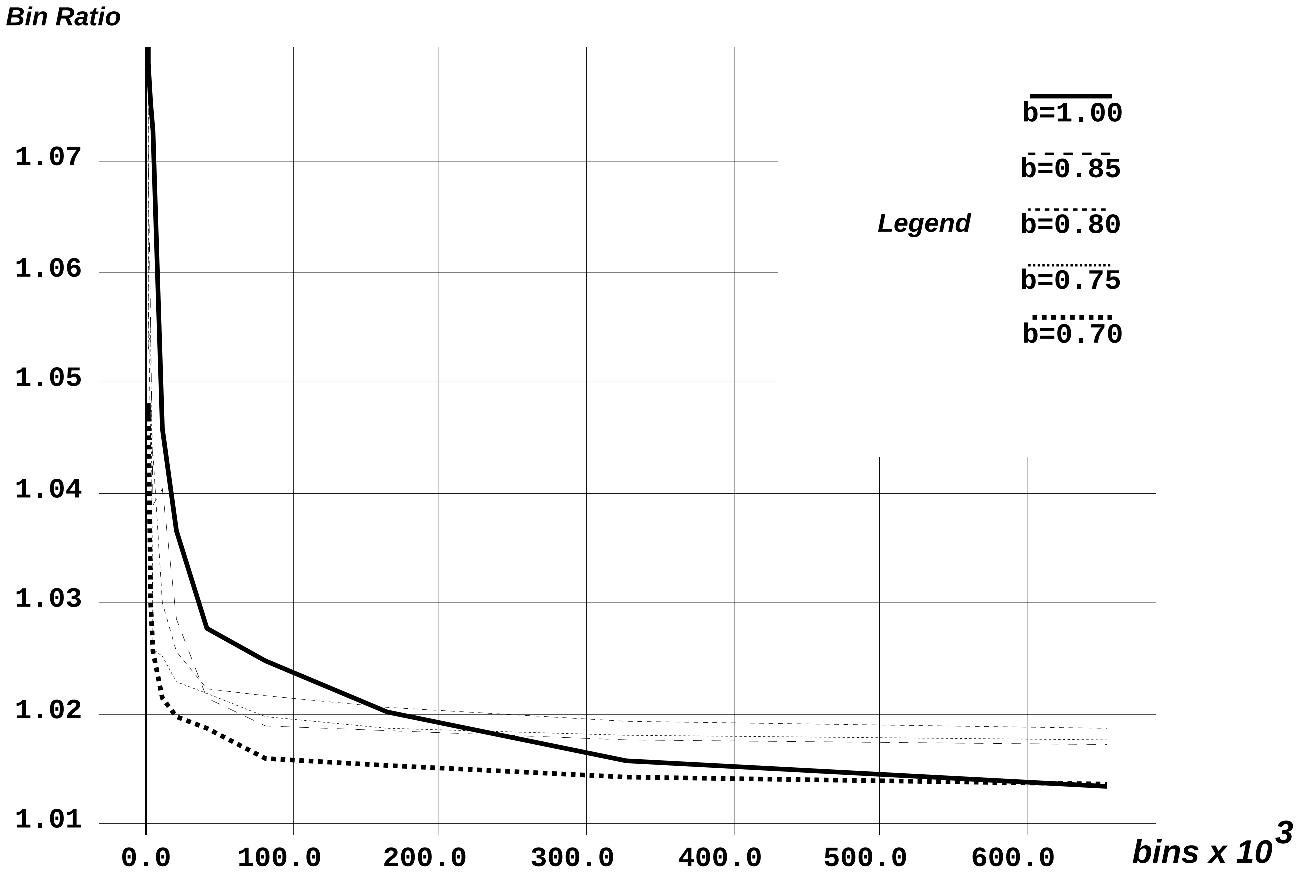
Efficiency measure: *Bin Ratio (or how does a heuristic compare to a perfect packing?)*

On-line heuristics: *First Fit, Best Fit, K-delayed Fitting (new and superior heuristic)*

Simulation method: *Hand-waving, Focussed hand-waving, Pipelined, Multi-pipelined*

Sample Exercise from a Lab Assignment: Bin Packing

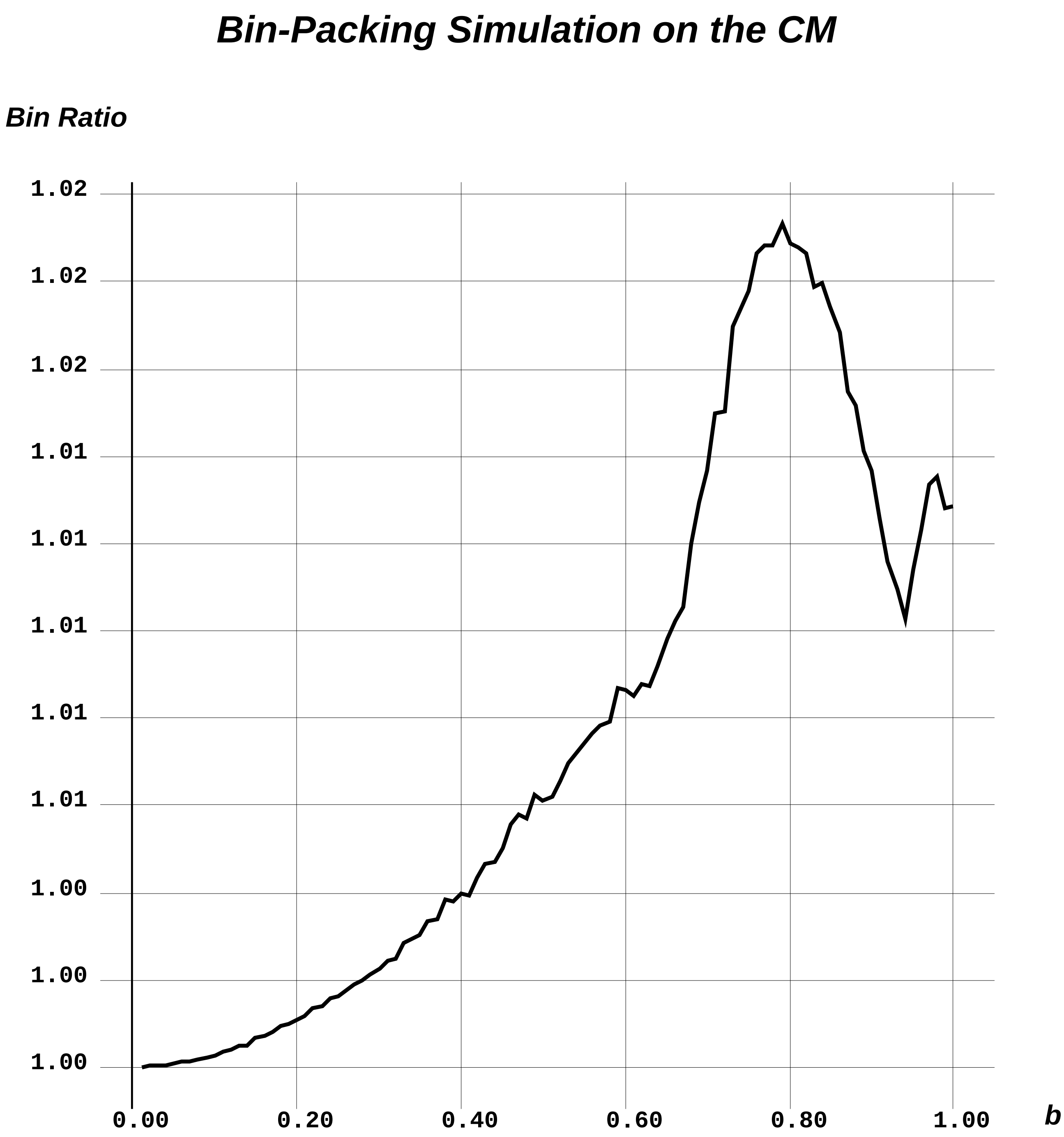
Bin-Packing Simulation on the CM



Bin Ratio Asymptotic Behavior

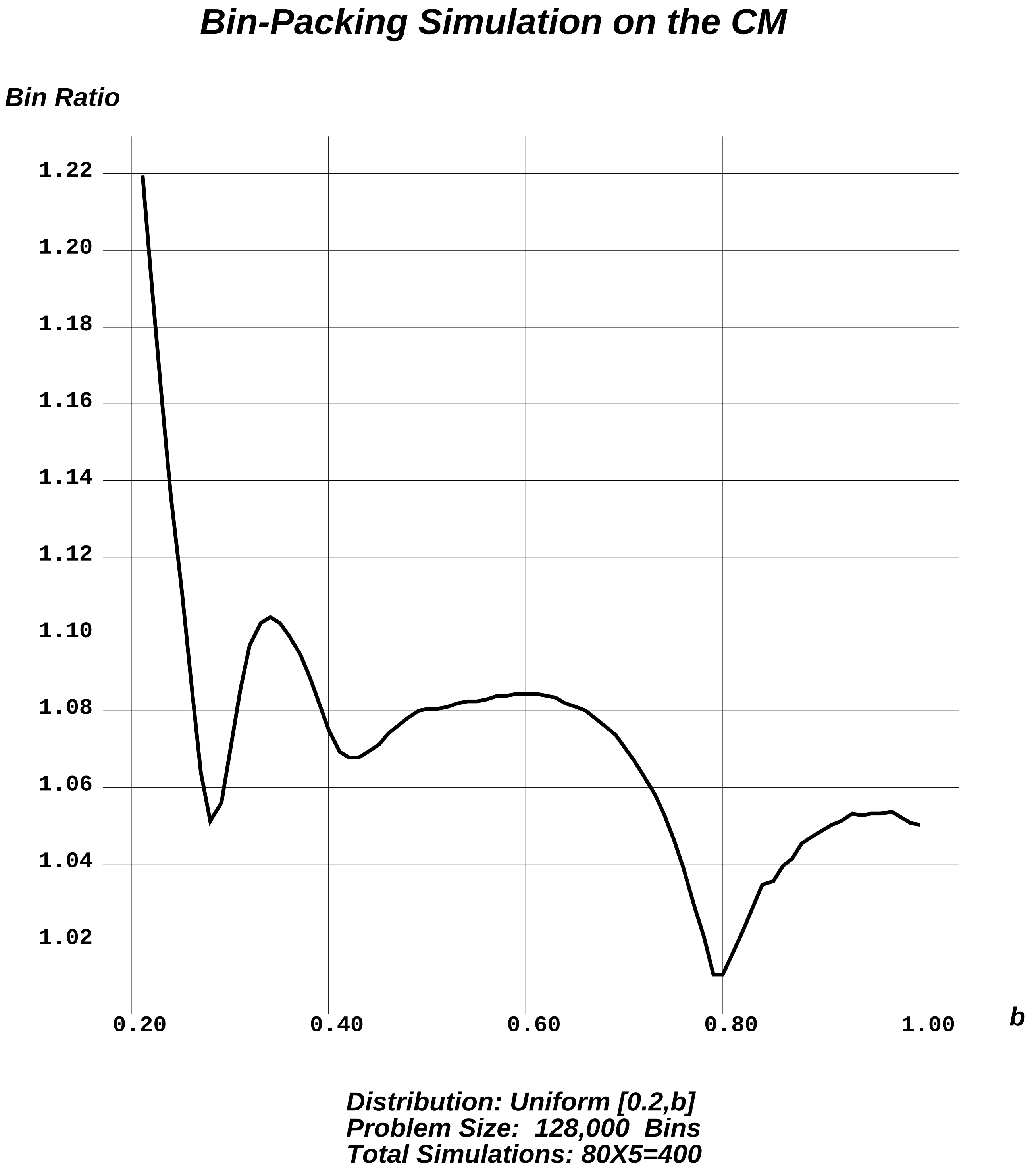
Distribution: Uniform [0.0,b]
Problem Size < 1 million Bins

Sample Exercise from a Lab Assignment: Bin Packing



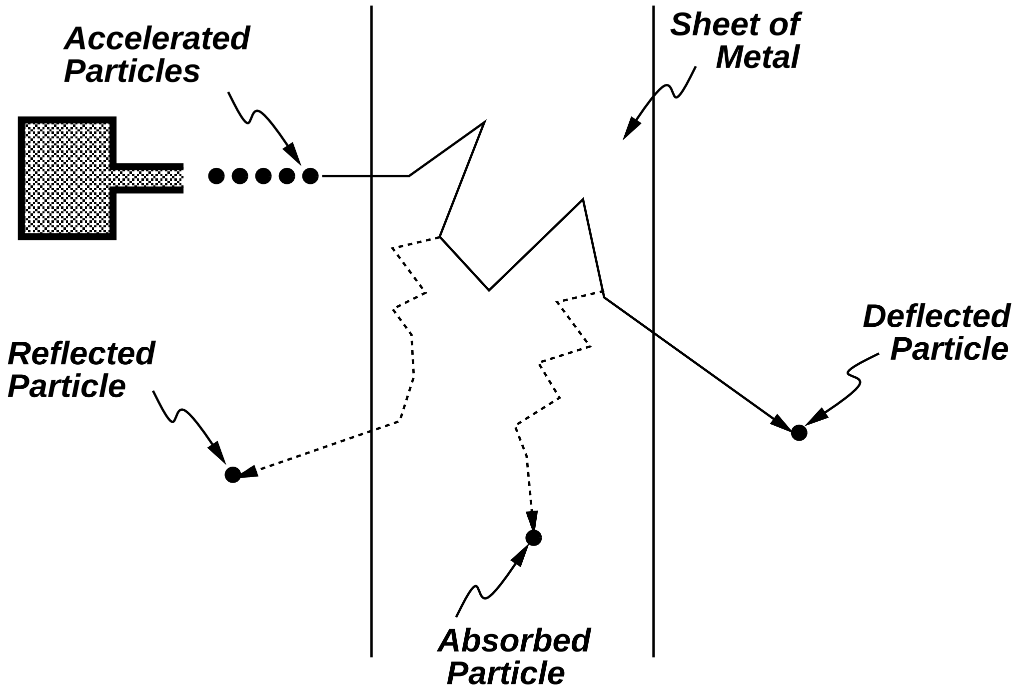
Distribution: Uniform [0.0,b]
Problem Size: 128,000 Bins
Total Simulations: 99X5=495

Sample Exercise from a Lab Assignment: Bin Packing



Sample Exercise from a Lab Assignment: Neutron Scattering

Efficient Monte-Carlo Simulation on the CM-2

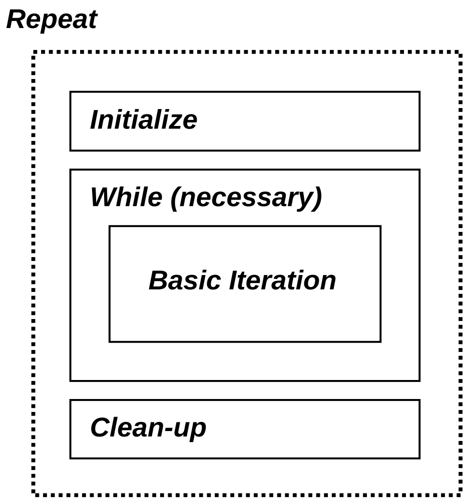


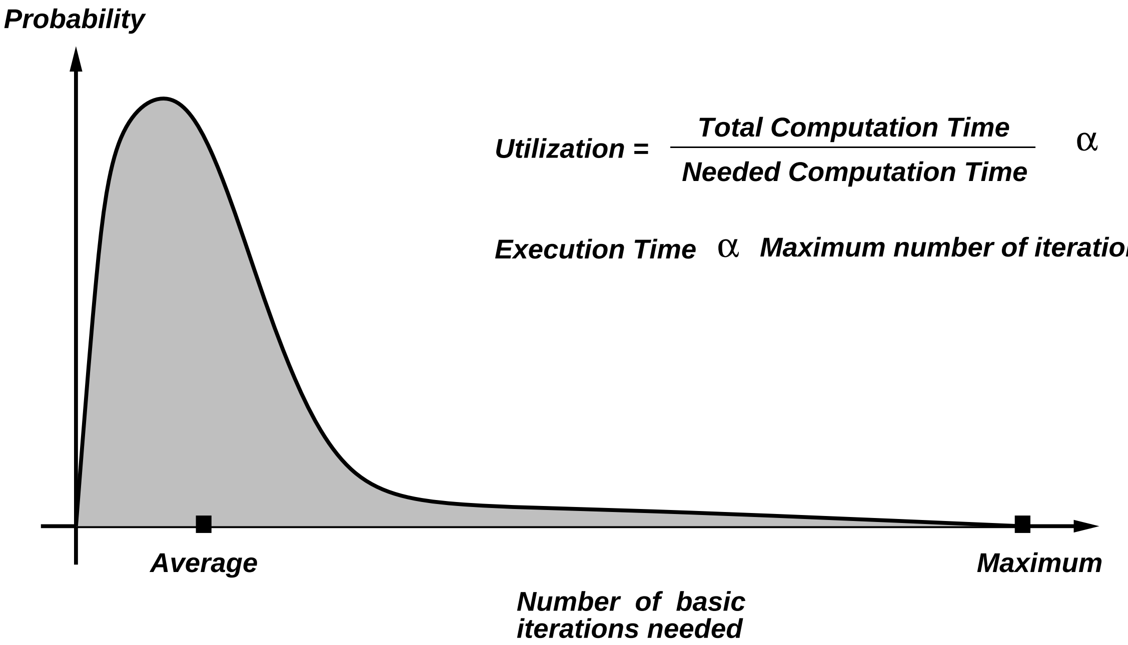
An example from particle physics

Sample Exercise from a Lab Assignment: Neutron Scattering

Typical CM-2 Monte-Carlo Simulation

```
While (More experiments needed)

  Initialize and activate all PEs ;

  While (Some PEs are still active)
    Perform an iteration in parallel ;
    If (Deflected/Reflected/Absorbed)
       Record state ;
       Deactivate ;

  Collect statistics ;
```





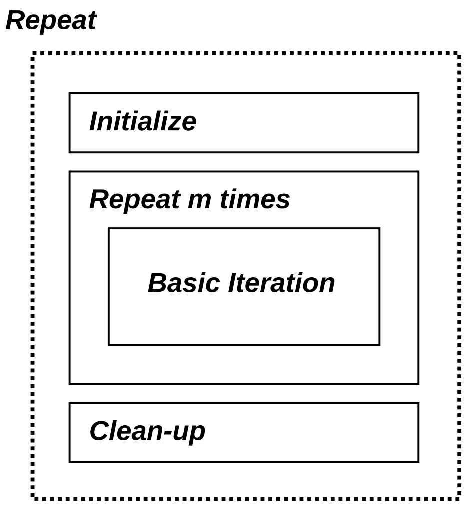
$$\textit{Utilization} = \frac{\textit{Total Computation Time}}{\textit{Needed Computation Time}} \propto \frac{1}{\textit{Range}}$$

$$\textit{Execution Time} \propto \textit{Maximum number of iteration}$$

Sample Exercise from a Lab Assignment: Neutron Scattering

Efficient CM-2 Monte-Carlo Simulation

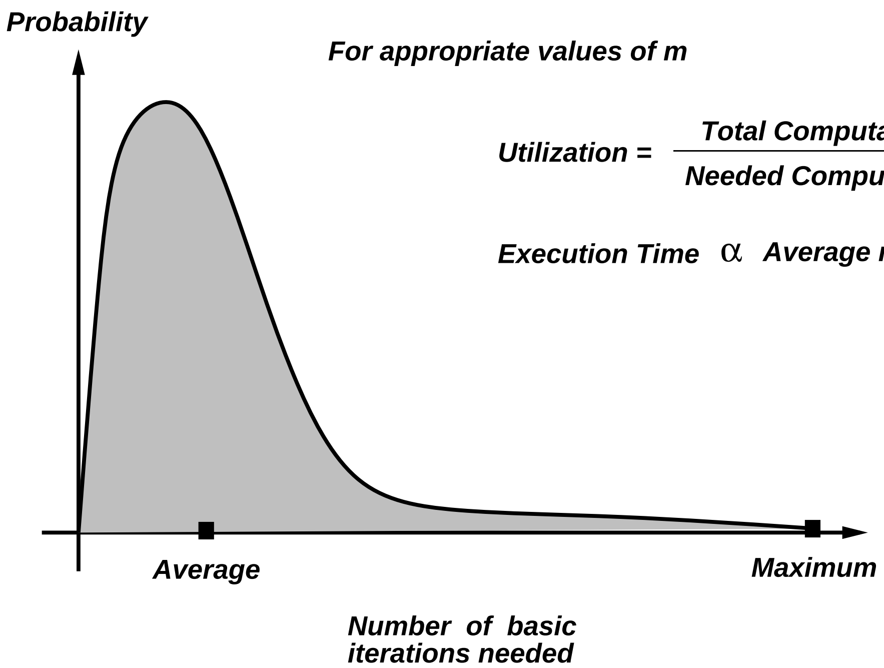
```
While (More experiments needed)

  Initialize and activate idle PEs ;

  Repeat m times
    Perform an iteration in parallel ;
    If (Deflected/Reflected/Absorbed)
       Record state ;
       Idle ;

  Collect statistics from idle PEs;
```



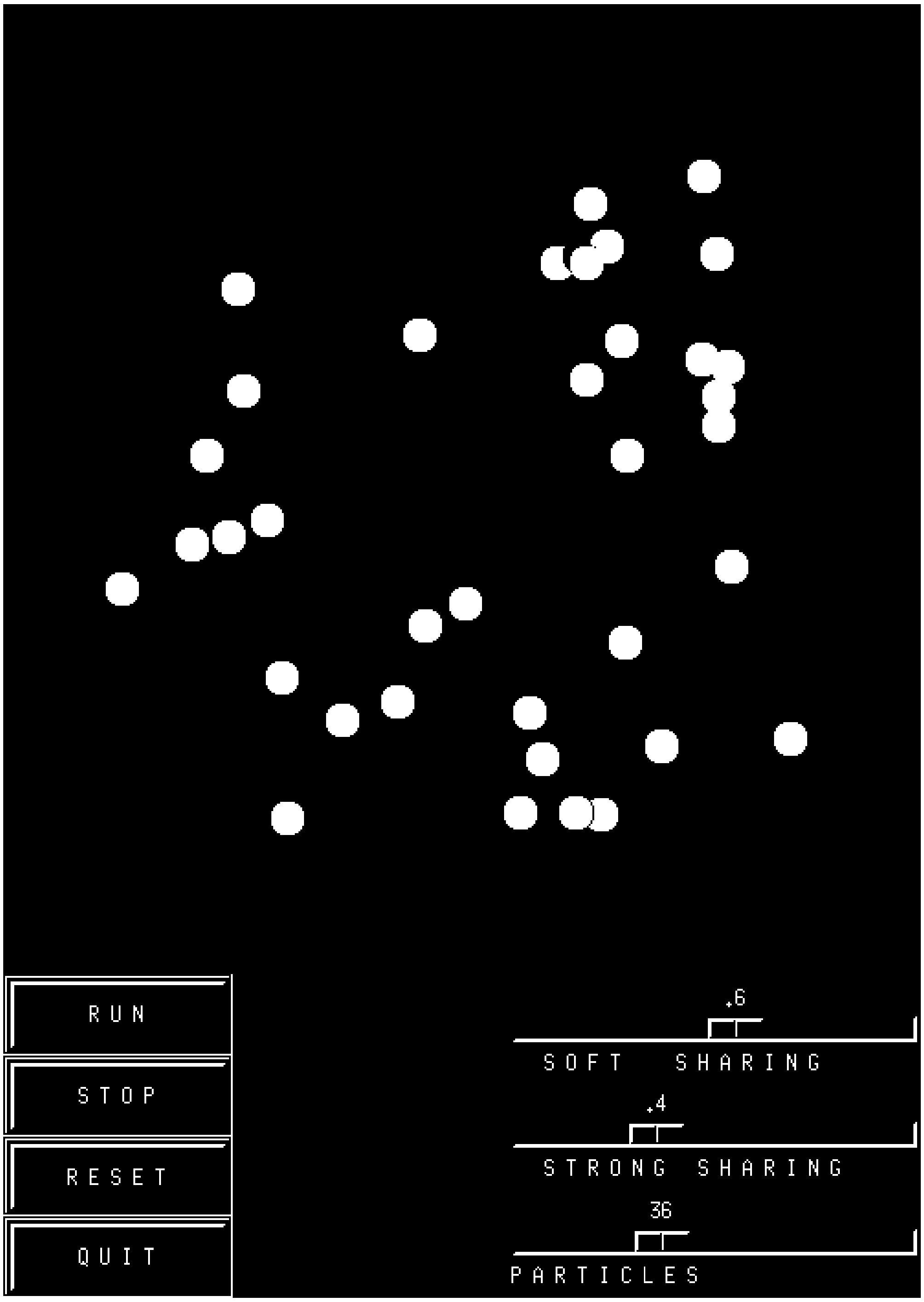
For appropriate values of m

$$\textit{Utilization} = \frac{\textit{Total Computation Time}}{\textit{Needed Computation Time}} \propto \frac{1}{SD}$$

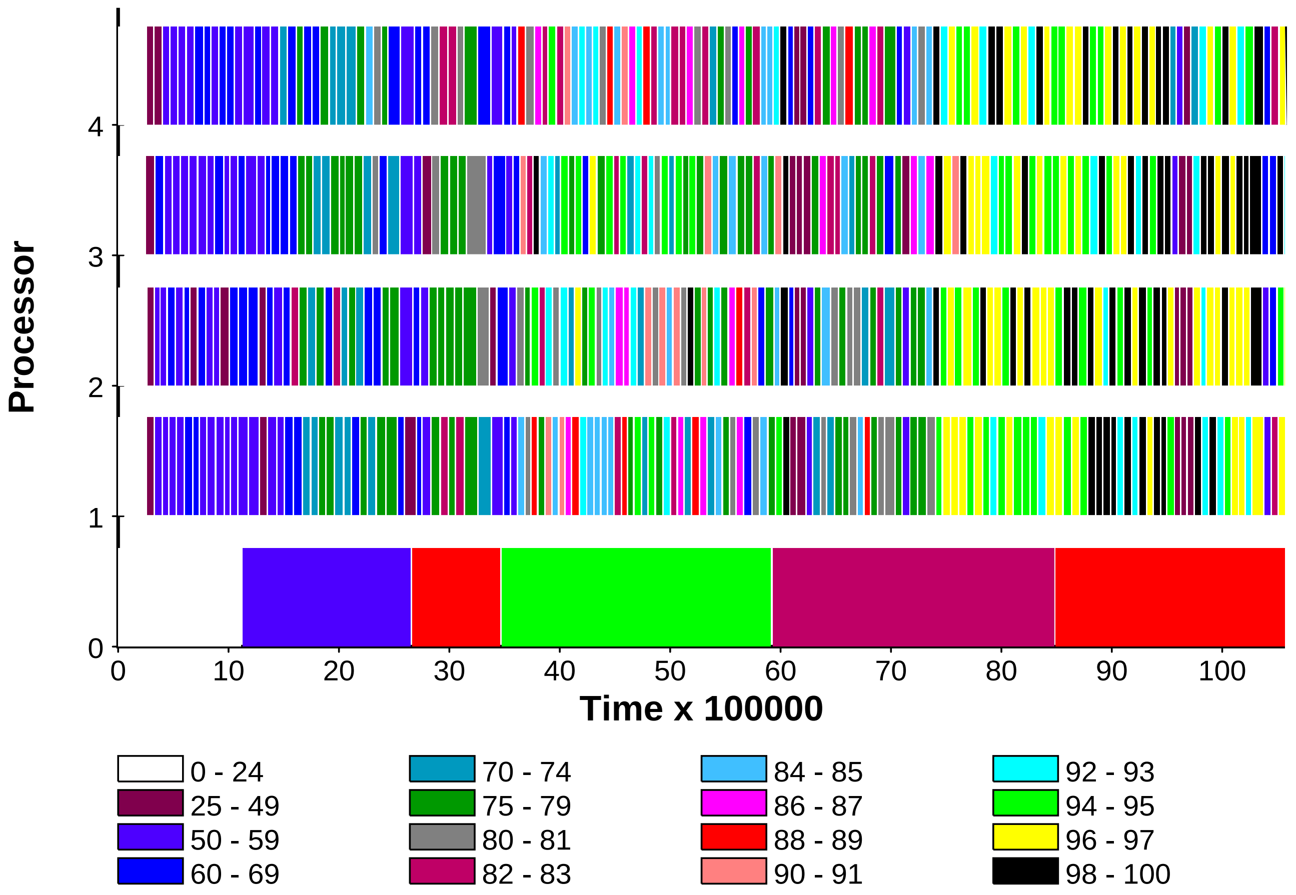
$$\textit{Execution Time} \propto \textit{Average number of iteration}$$



Sample Exercise from a Lab Assignment: Snoopy Caching



Sample Exercise from a Lab Assignment: Snoopy Caching



Sample Exercise from a Lab Assignment: Snoopy Caching