

ECE 566- Parallel and Distributed Computing

Lecture 1: Introduction

Manish Parashar

parashar@ece.rutgers.edu

Department of Electrical & Computer
Engineering

Rutgers University

ECE 566 - Objectives

- ◆ To study the theory and practice of applied parallel/distributed computing
 - current state of parallel/distributed architectures and systems
 - issues and approaches in parallel/distributed application development
 - parallel/distributed programming
 - current and future trends

ECE 566 - Tentative Schedule

- ◆ Introduction
 - » 1 Lecture
- ◆ Architectures/Issues/Applications
 - » 4 Lectures
- ◆ Measuring Performance of P/D Archs./Apps.
 - » 1 Lecture
- ◆ Introduction P/D Programming
 - » 1 Lecture
- ◆ Message Passing
 - » 4 Lectures
- ◆ Shared Memory
 - » 4 Lectures

ECE 566 - Tentative Schedule

- ◆ Multithreading
 - » 2 Lecture
- ◆ Distributed Object
 - » 4 Lectures
- ◆ Cluster Computing
 - » 4 Lectures
- ◆ Meta-computing
 - » 2 Lecture
- ◆ Enterprise Computing
 - » 2 Lectures

ECE 566 - Grading Policy

- ◆ Programming Projects (3 * 15% = 45%)
 - » You will be assigned 3 programming projects. An approximate schedule is
 - ◆ Message passing programming - due end of February
 - ◆ Shared memory programming - due after spring break
 - ◆ Distributed object programming - due mid April.
- ◆ Midterm Exam (25%) - 03/09/00
- ◆ Final Project (30%) - Due 04/25/00

ECE 566 - Recommended Readings

- ◆ *Scalable Parallel Computing* K. Hwang & Z. Xu, WCB/McGraw-Hill, 1998.
- ◆ *High Performance Cluster Computing Volume I & II* Rajkumar Buyya, Prentice Hall, 1999.
- ◆ *Parallel Computer Architecture: A Hardware/Software Approach* D. Cullur, J.P. Singh, A. Gupta, Morgan Kaufmann Publishers, 1998.
- ◆ *In Search of Cluster: The Ongoing Battle In Lowly Parallel Computing* G.F. Pfister, Prentice Hall, 1998.
- ◆ *Java Distributed Computing* Jim Farley, O'Reilly Publishers, 1998.
- ◆ *The Essential Distributed Objects Survival Guide*, Robert Orfali, Dan Harkey, and Jeri Edwards, John Wiley & Sons, Inc., 1994.
- ◆ *Parallel Programming with MPI* P. Pacheco, Morgan Kaufmann Publishers, 1996.
- ◆ *Designing and Building Parallel Programs* I. Foster, Addison-Wesley, 1995.

ECE 566 - Other Information

- ◆ TA:
 - » Ashish Gaini (ashishg@caip.rutgers.edu)
- ◆ My Office Hours:
 - » Tuesday/Thursday 9:00 - 10:00 AM, CoRE 504
- ◆ TA Office Hours:
 - » Wednesday 11:00 AM - 1:00 PM, CoRE 509
- ◆ You can email me at
parashar@caip.rutgers.edu

ECE 566 - What you need to do...

- ◆ Email contact information
 - » Name
 - » SS#
 - » Major
 - » Research
 - » Contact #
- ◆ Get computer account on the ece/caip systems

How to Run App. Faster ?

- ◆ There are 3 ways to improve performance:
 - » 1. Work Harder
 - » 2. Work Smarter
 - » 3. Get Help
- ◆ Computer Analogy
 - » 1. Use faster hardware: e.g. reduce the time per instruction (clock cycle).
 - » 2. Optimized algorithms and techniques, customized hardware
 - » 3. Multiple computers to solve problem: That is, increase no. of instructions executed per clock cycle.

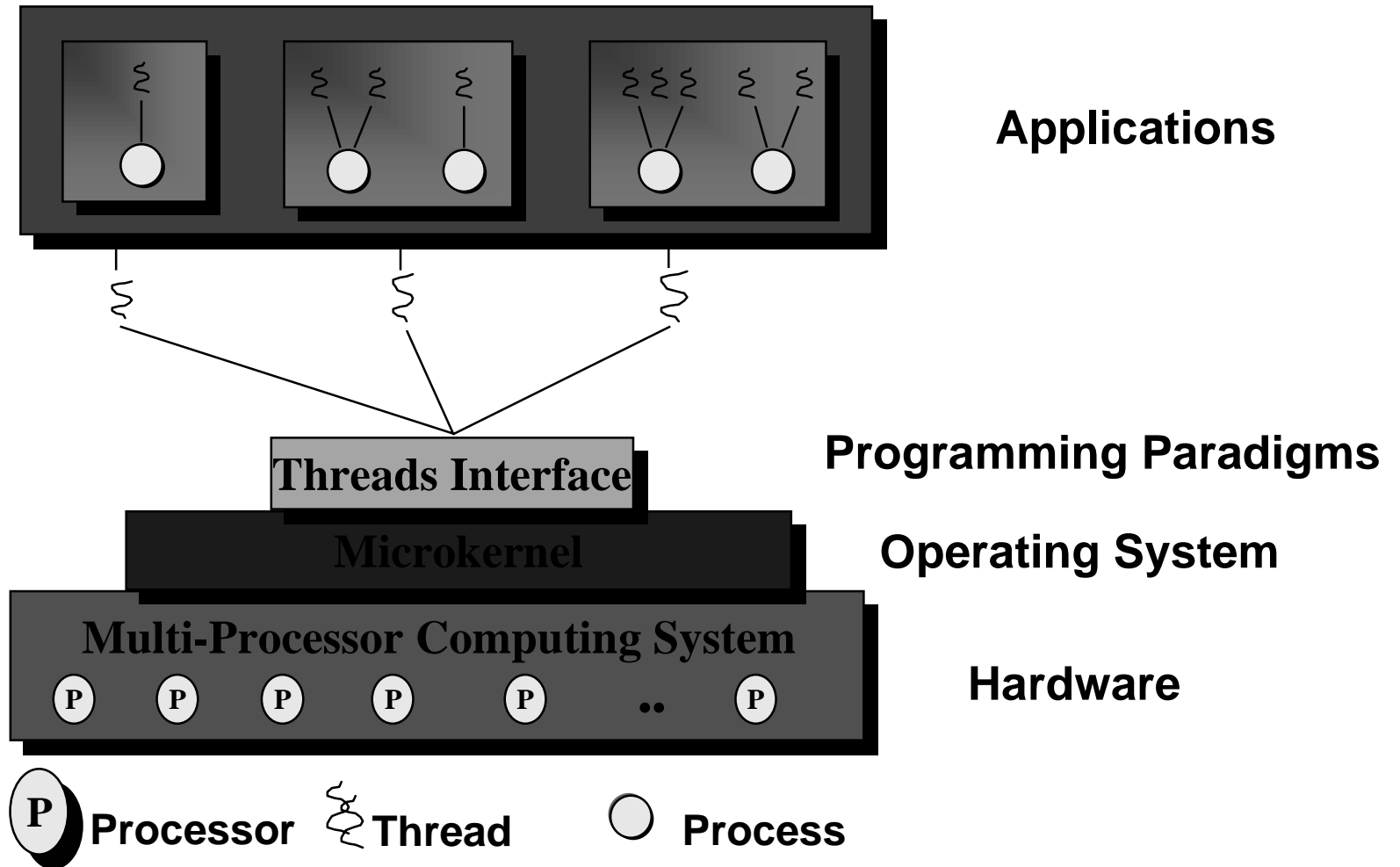
Goals and Motivations

- ◆ Increased “quality”
 - availability, accessibility, serviceability
- ◆ High performance
 - greater than the fastest uniprocessor
- ◆ Competitive cost-performance
- ◆ Scalable
- ◆ Practical
- ◆ Usable

Scalable ?

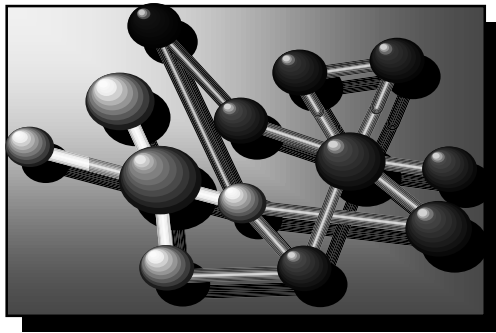
- ◆ Ts/Tp
- ◆ Issues
 - “.... is it parallelizable ?”
 - communication bandwidth
 - memory bandwidth (distributed memory)
 - IO bandwidth
- ◆ Amdahl's law....

Computing Elements

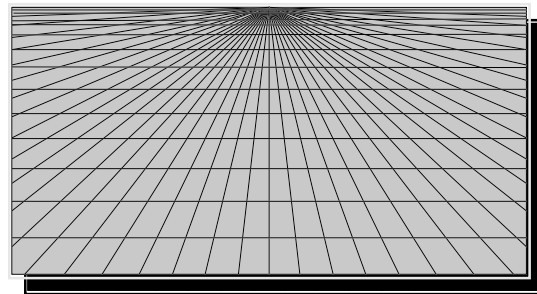


Need of more Computing Power: Grand Challenge Applications

- ◆ Solving technology problems using
- ◆ computer modeling, simulation and analysis



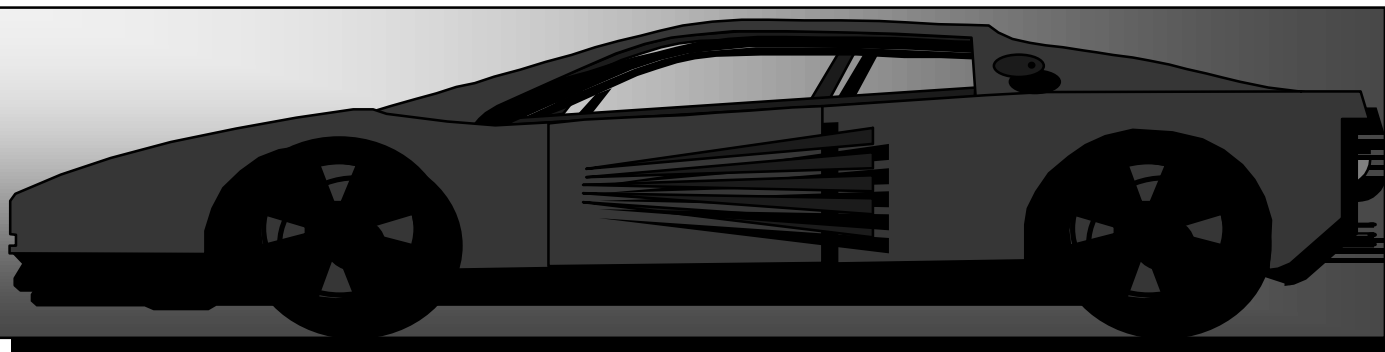
Life Sciences



Aerospace



Geographic
Information
Systems

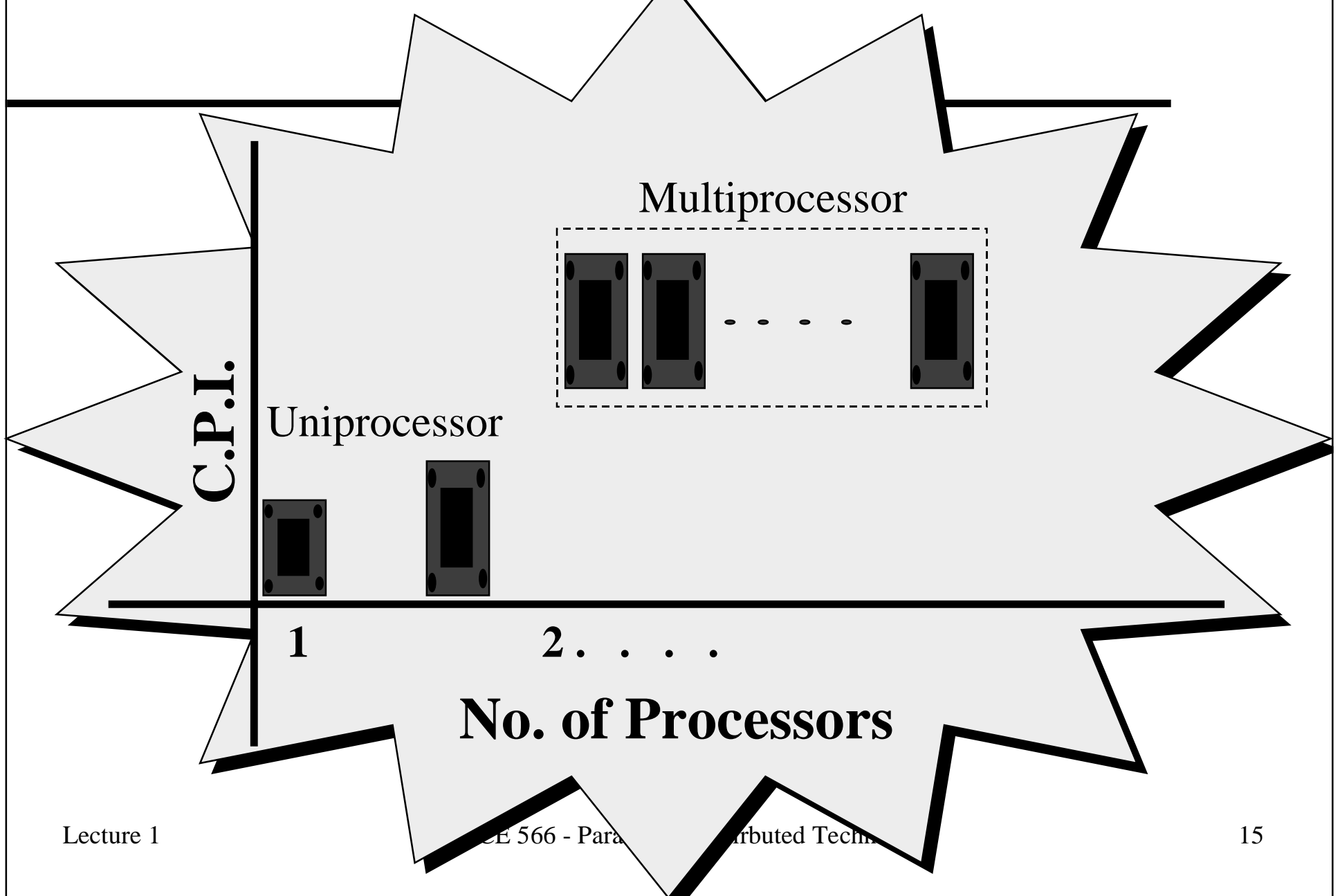


Mechanical Design & Analysis (CAD/CAM)

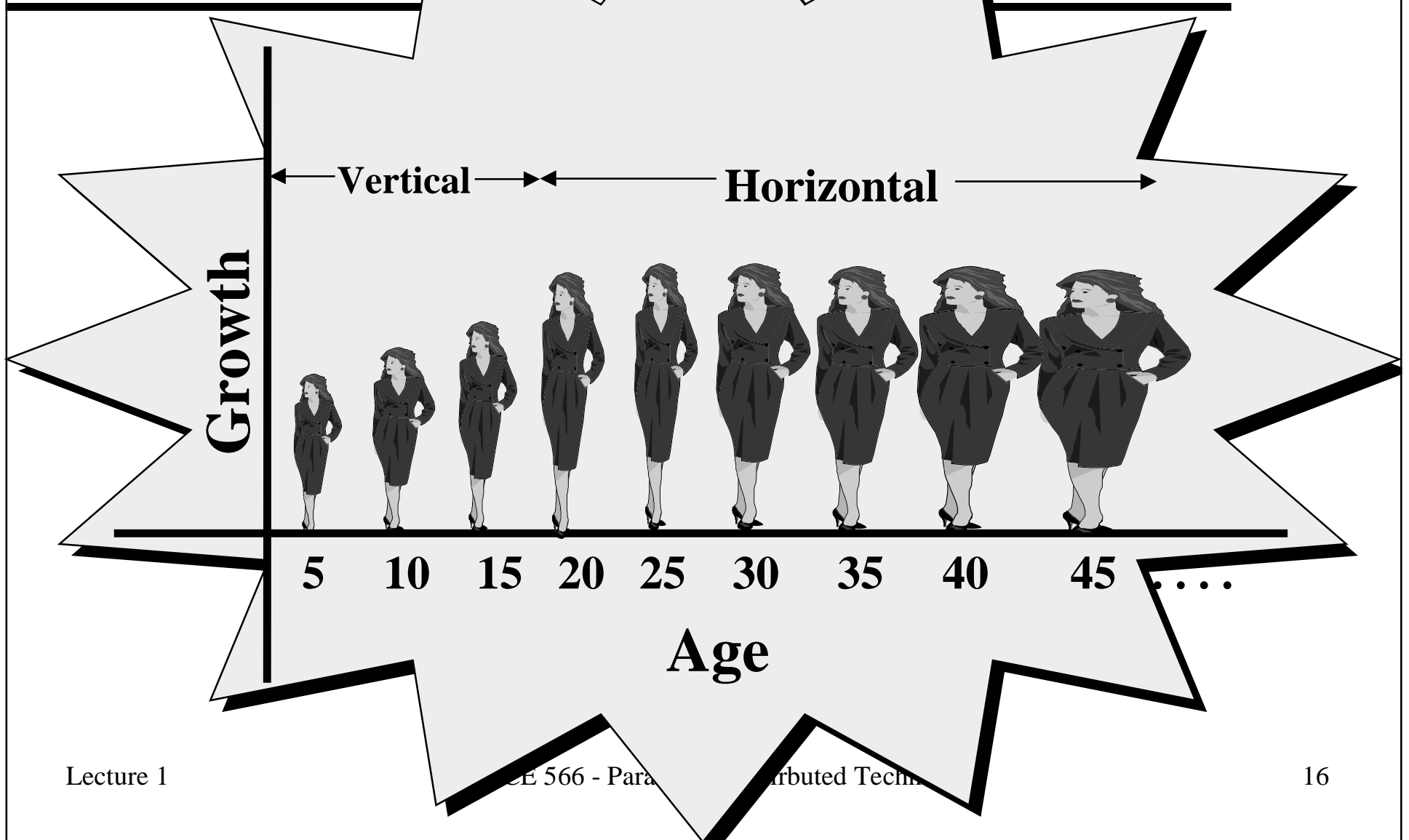
Sequential Architecture Limitations

- Sequential architectures reaching physical limitation (speed of light, thermodynamics)
- Hardware improvements like pipelining, Superscalar, etc., are non-scalable and requires sophisticated Compiler Technology.
- Vector Processing works well for certain kind of problems.

Computational Power Improvement



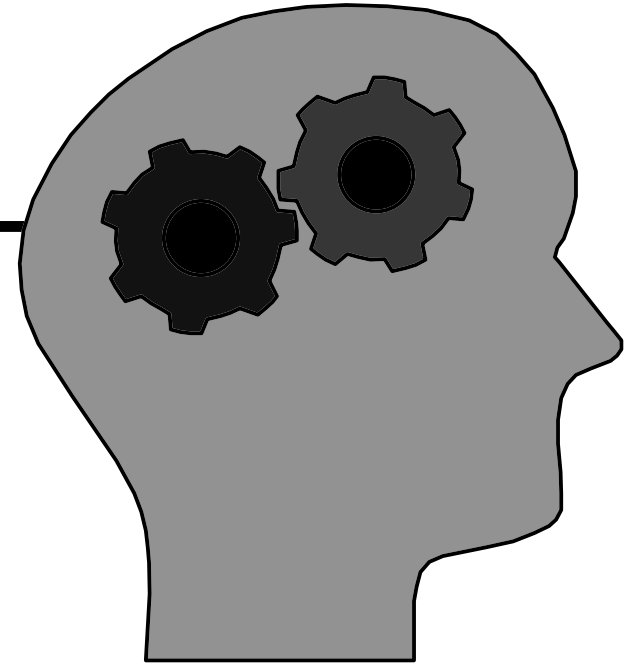
Human Physical Growth Analogy: Computational Power Improvement



Why Parallel/Distributed Processing NOW?

- ◆ The Tech. of PP is mature and can be exploited commercially; significant R & D work on development of tools & environment.
- ◆ Significant development in Networking technology is paving a way for heterogeneous computing.

Motivating Factors



- ◆ Aggregated speed with which complex calculations carried out by millions of neurons in human brain is amazing! although individual neurons response is slow (msec.) - demonstrate the feasibility of PP

Architecture Spectrum

- ◆ Shared-Everything
 - Symmetric Multiprocessors
- ◆ Shared Memory
 - NUMA, CC-NUMA
- ◆ Distributed Memory
 - DSM, Message Passing
- ◆ Shared-Nothing
 - Clusters, NOW's
- ◆ Client/Server

Pros and Cons

◆ Shared Memory

– Pros

- ◆ flexible, easier to program

– Cons

- ◆ not scalable, synchronization/coherency issues

◆ Distributed Memory

– Pros

- ◆ scalable

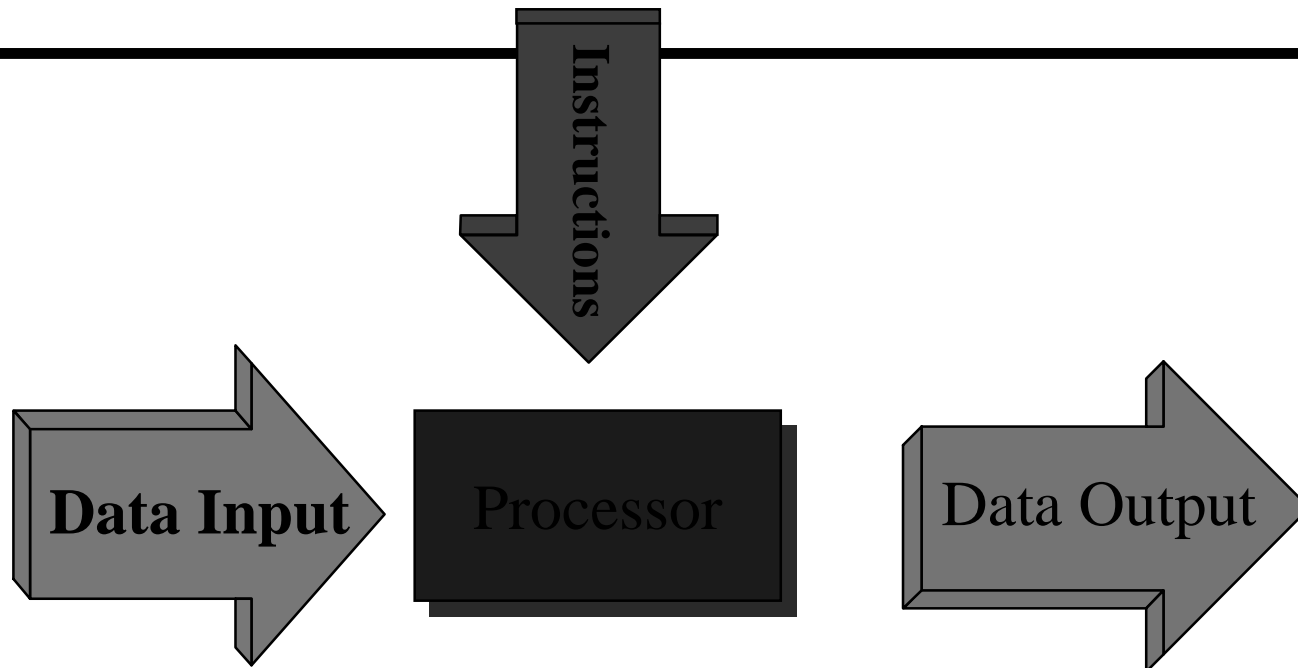
– Cons

- ◆ difficult to program, require explicit message passing

Taxonomy of Architectures

- ◆ Simple classification by Flynn (1967):
- ◆ (No. of instruction and data streams)
 - SISD - conventional
 - SIMD - data parallel, vector computing
 - MISD - systolic arrays
 - MIMD - very general, multiple approaches.
- ◆ Current focus is on MIMD model, using general purpose processors or multicomputers.

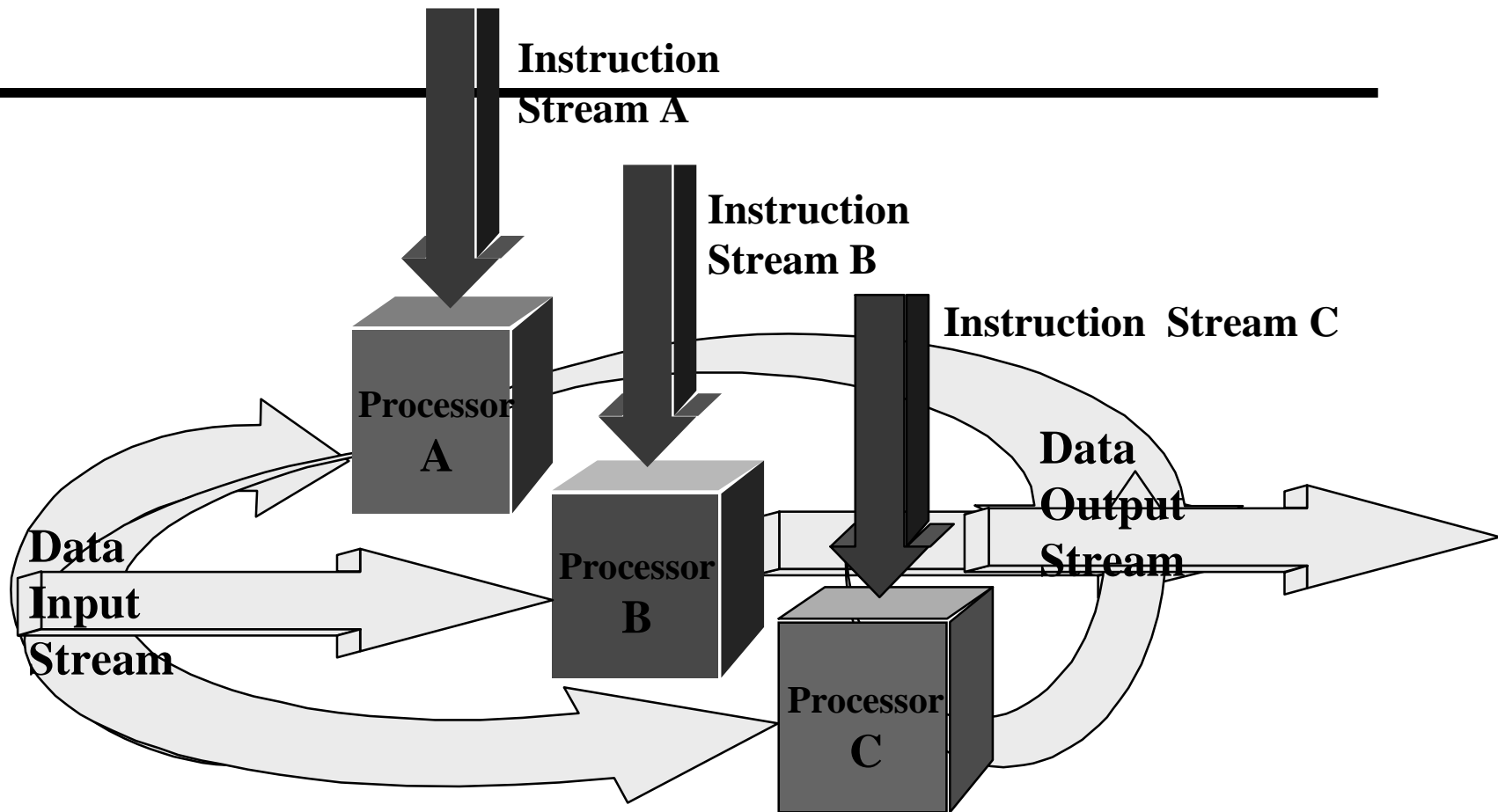
SISD : A Conventional Computer



→ Speed is limited by the rate at which computer can transfer information internally.

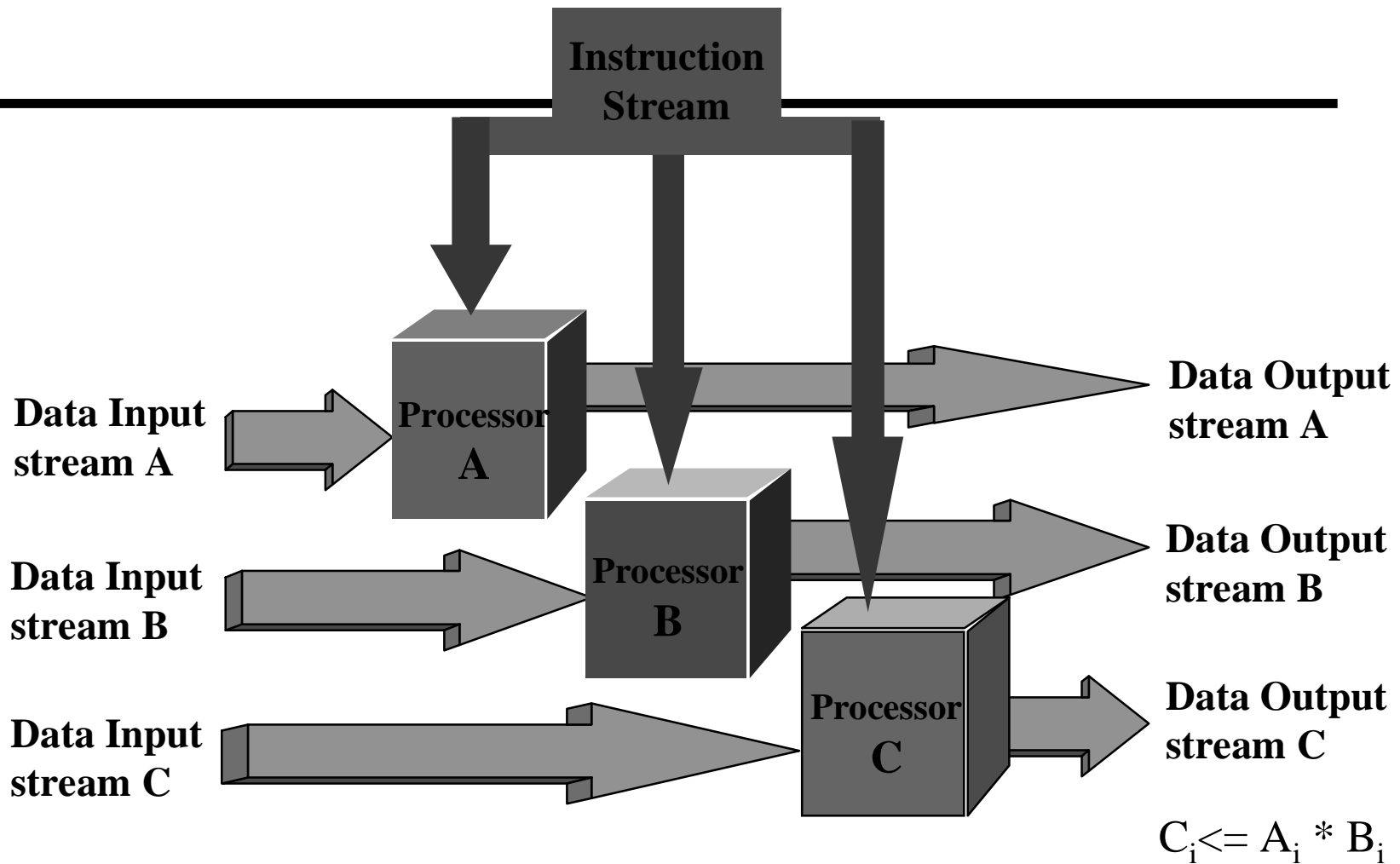
Ex:PC, Macintosh, Workstations

The MISD Architecture



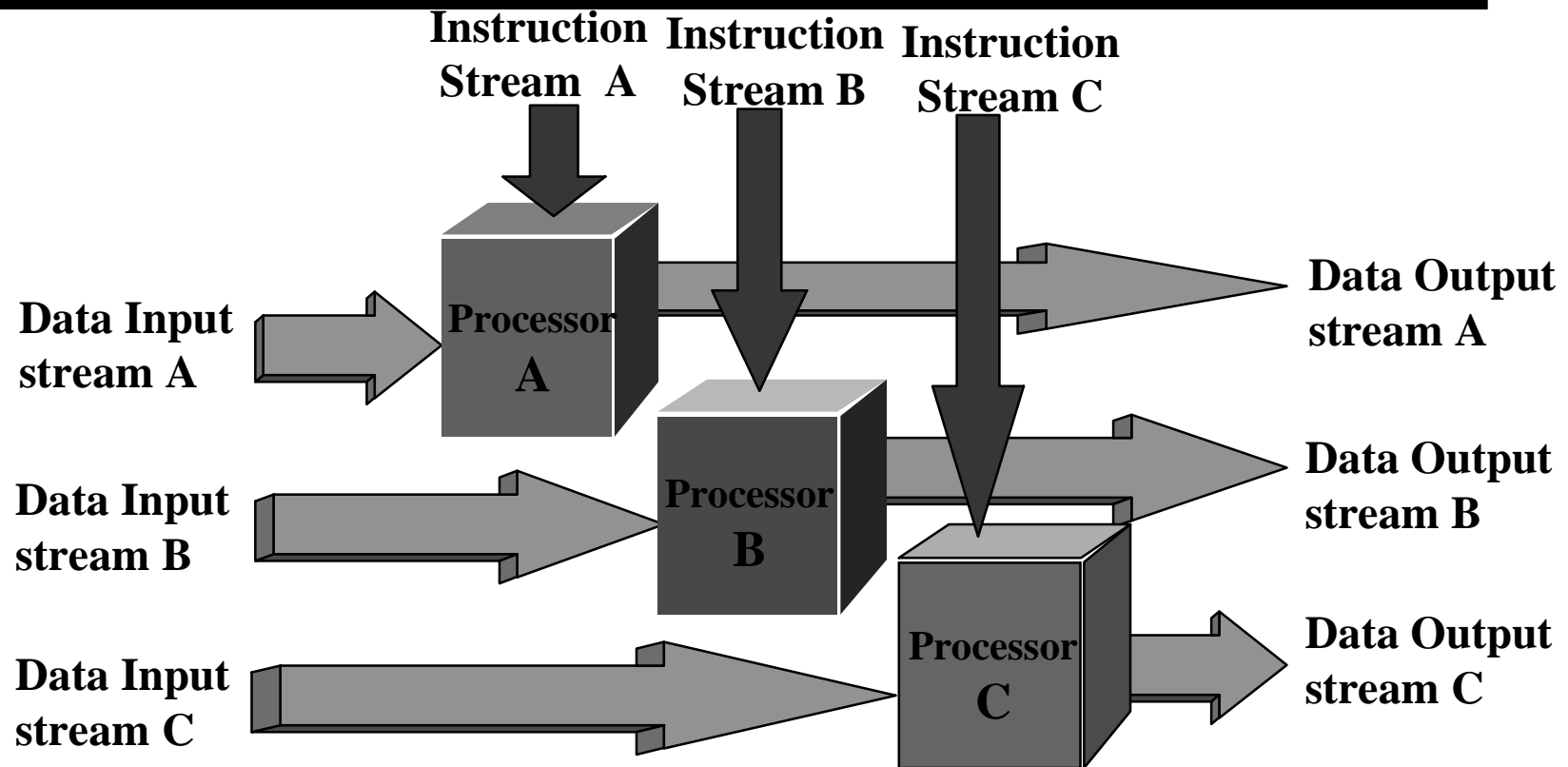
→ More of an intellectual exercise than a practical configuration. Few built, but commercially not available

SIMD Architecture



Ex: CRAY machine vector processing, Thinking machine cm*

MIMD Architecture

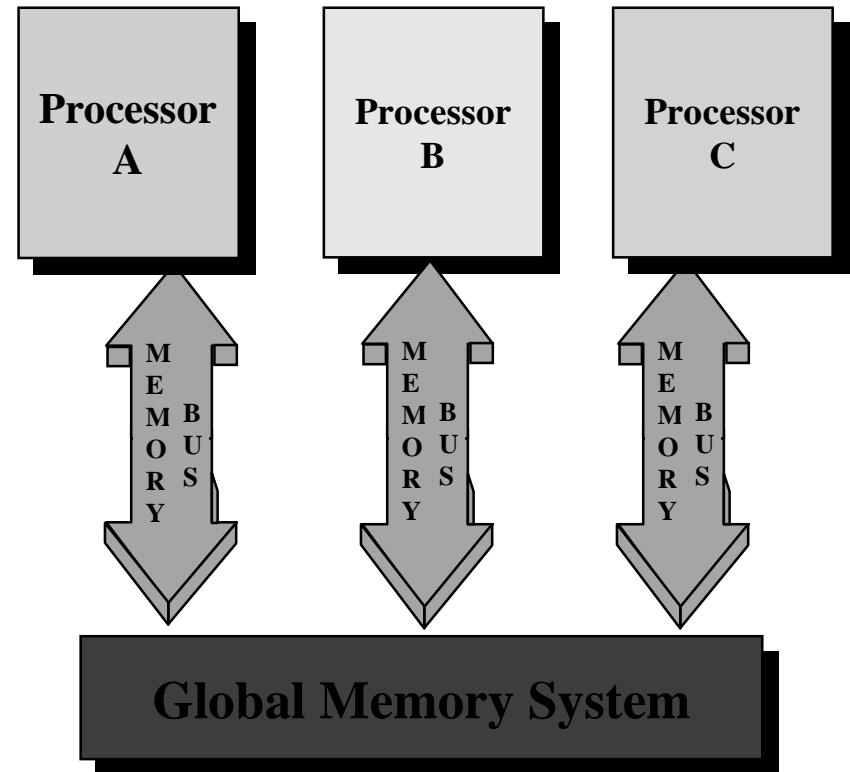


- ◆ Unlike SISD, MISD, MIMD computer works asynchronously.
 - » Shared memory (tightly coupled) MIMD
 - » Distributed memory (loosely coupled) MIMD

Shared Memory MIMD machine

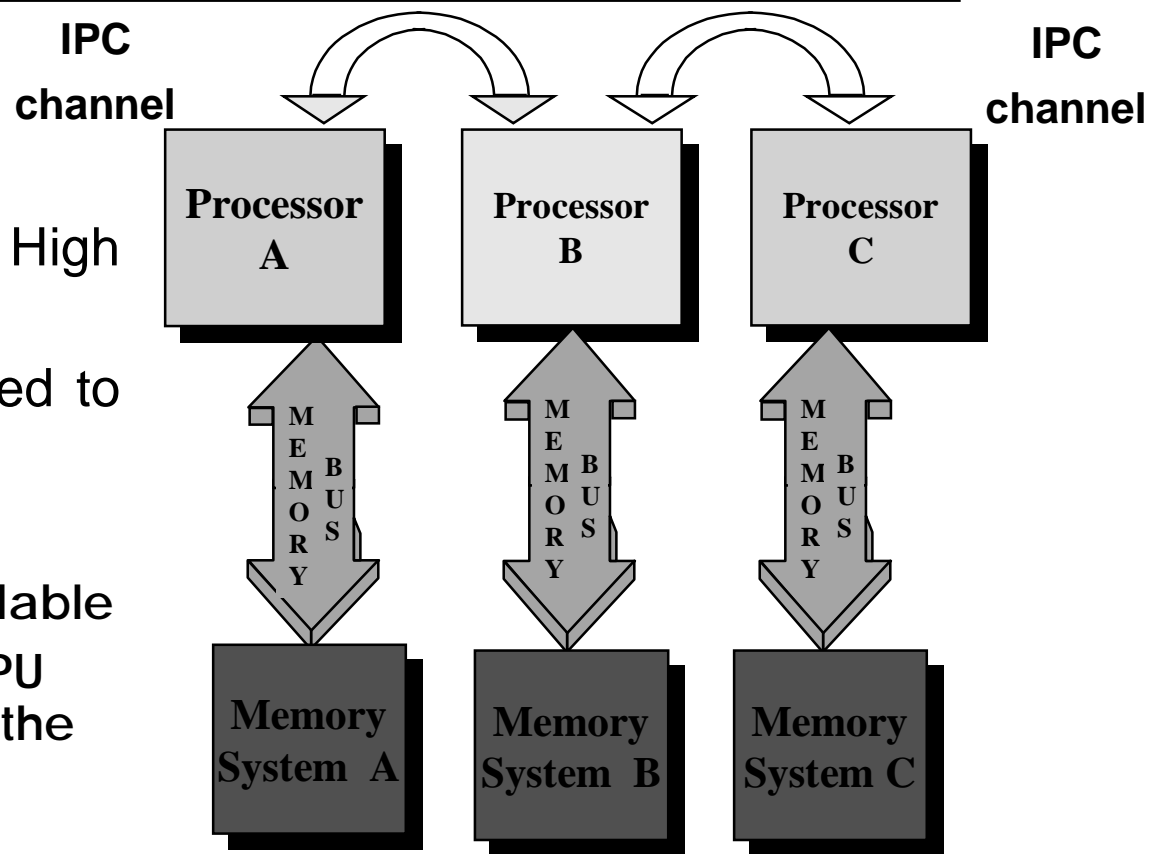
Comm: Source PE writes data to GM & destination retrieves it

- Easy to build, conventional OSES of SISD can be easily be ported
- Limitation : reliability & expandability.
A memory component or any processor failure affects the whole system.
- Increase of processors leads to memory contention.
Ex. : Silicon graphics supercomputers....



Distributed Memory MIMD

- Communication : IPC on High Speed Network.
- Network can be configured to ... Tree, Mesh, Cube, etc.
- Unlike Shared MIMD
 - easily/ readily expandable
 - Highly reliable (any CPU failure does not affect the whole system)



Main HPC Architectures

- ◆ SISD - mainframes, workstations, PCs.
- ◆ SIMD Shared Memory - Vector machines, Cray...
- ◆ MIMD Shared Memory - Sequent, KSR, Tera, SGI, SUN.
- ◆ SIMD Distributed Memory - DAP, TMC CM-2...
- ◆ MIMD Distributed Memory - Cray T3D, Intel, Transputers, TMC CM-5, plus recent workstation clusters (IBM SP2, DEC, Sun, HP).

Main HPC Architectures

- ◆ NOTE: Modern sequential machines are not purely SISD - advanced RISC processors use many concepts from
 - vector and parallel architectures (pipelining, parallel execution of instructions, prefetching of data, etc) in order to achieve one or more arithmetic operations per clock cycle.

Parallel Processing Paradox

- ◆ Time required to develop a parallel application for solving GCA is equal to:
 - Half Life of Parallel Supercomputers.

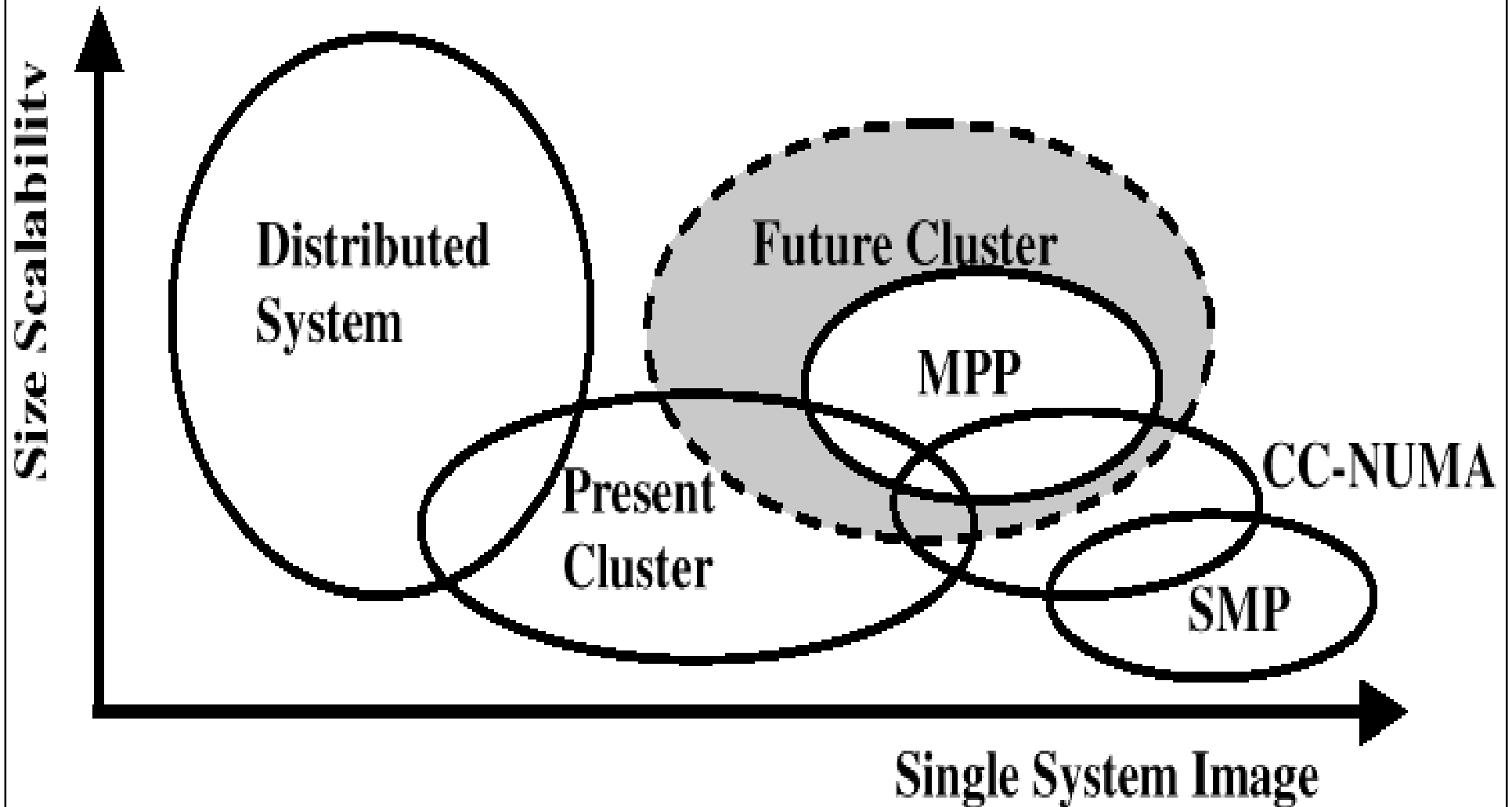
An Alternative Supercomputing Resources

- ◆ Vast numbers of under utilised workstations available to use.
- ◆ Huge numbers of unused processor cycles and resources that could be put to good use in a wide variety of applications areas.
- ◆ Reluctance to buy Supercomputer due to their cost and short life span.
- ◆ Distributed compute resources “fit” better into today's funding model.

Scalable Parallel Computers

Characteristic	MPP	SMP, CC-NUMA	Cluster	Distributed System
Number of Nodes	$O(100) - O(1000)$	$O(10) - O(100)$	$O(100)$ or less	$O(10) - O(1000)$
Node Complexity	Fine or medium grain	Medium or coarse grain	Medium grain	Wide range
Internode Communication	Message passing or shared variables for DSM	Centralized and distributed shared memory	Message Passing	Shared files, RPC, message passing, IPC protocol
Job Scheduling	Single run queue at host	Single run queue mostly	Multiple queues but coordinated	Independent multiple Queues
SSI Support	Partially	Always in SMP and some NUMA	Desired	No
Node OS Copies and Type	N microkernels and 1 monolithic OS at host	One monolithic for SMP and multiple for NUMA	N OS platforms (Homogeneous or microkernel)	N OS platforms (Heterogeneous)
Address Space	Multiple (Single for DSM)	Single	Multiple or single	Multiple
Internode Security	Unnecessary	Unnecessary	Required if exposed	Required
Ownership	One organization	One organization	One or More organizations	Many organizations
Network Protocol	Nonstandard	Nonstandard	Standard & Lightweight	Standard

Design Space of Competing Computer Architecture



Acknowledgement

- ◆ High Performance Cluster Computing
 - Rajkumar Buyya (rajkumar@ieee.org)