Why are we building parallel systems?

- Single processor performance
  - Transistor density
  - IC
  - Speed
  - ILP
- Speed increases, power increases, heat increases
- Chips too hot to be reliable. Limits to heat dissipation
- How to exploit the increasing number of transistor density?
  - Add parallelism
  - Go multicore

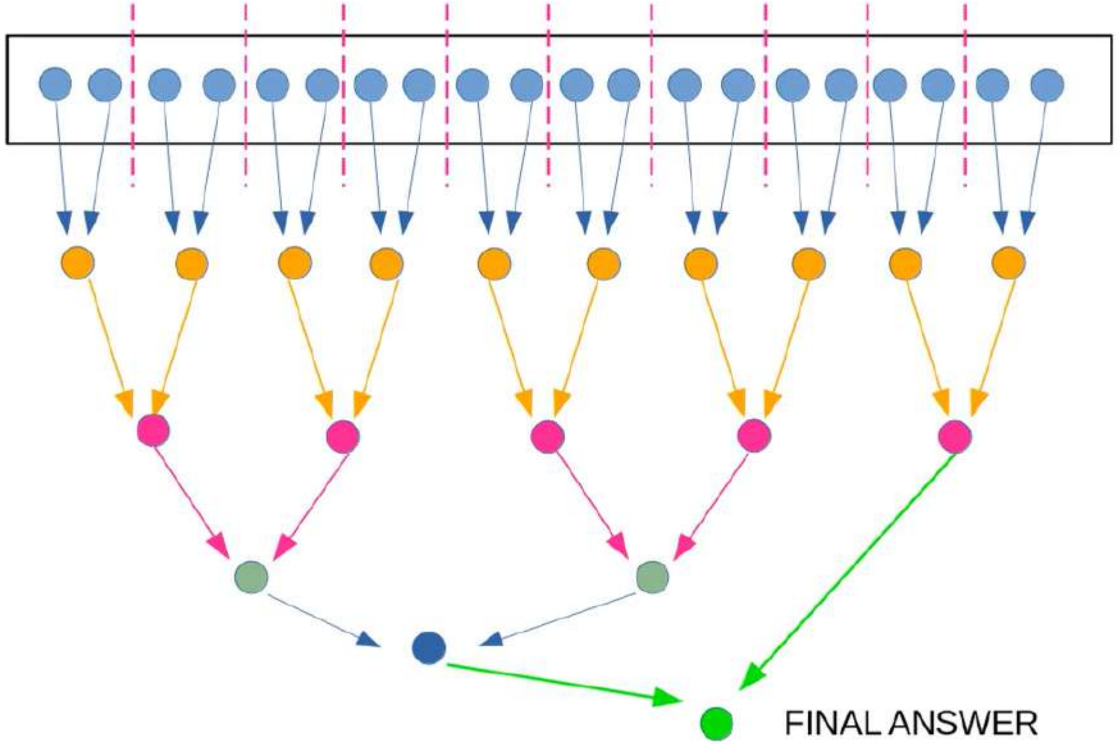
Why we need to write parallel programs?

- Programs written for conventional single core systems cannot exploit presence of multiple cores
- We can run multiple instances of a program on a multicore system. BUT this is of less help
- e.g. running multiple instances of gaming program? Is not desirable !
  - We want the game to run faster !
- Rewrite serial programs to parallel programs so that they use multiple cores
- Rewrite or translate serial code
  - Translate => can we auto-translate?
  - Till now researchers have had limited success in writing programs that convert the serial program to parallel program
  - It is not straightforward to convert serial to parallel by identifying par-constructs
- Rather, step back and devise entirely new algorithm

+42 NS MK RM VA UK AM VP VK

VARHADE AMEY ANANT UJWAL KUMAR ANSHUL MITTAL VIJAY PUROHIT VEDIKA JITENDRA KULKARNI RATHOD SAINATH Hemangee Kalpesh Kapoor

Example: sum of array elements



Example

- Sum of all elements of an array
- Tough for the translator program to discover the procedure
- Certain softwares can identify common serial constructs and efficiently parallelise them
- HOWEVER to apply the principles on ever increasing complex serial programs is difficult
- Therefore, we cannot simply continue to write serial programs.
- We MUST write parallel programs to exploit the power of multiple cores

How do we write parallel programs

- Basic idea is of partitioning the work to be done among the cores
  - Task parallelism
  - Data parallelism
- Ex: Prof. P, TAs: A, B, C, D
  - Have to check 100 answer books with 5 questions each
  - Data parallel => P, A, B, C, D each take 20 copies to check
  - Task parallel => P, A, B, C, D each take 1 question each to check

+52 VP VA UK AM IMLIJUNGLA LONGCHAR VK VEDIKA JITENDRA KULKARNI RATHOD SAINATH BHASKER GOEL VB VANSHITA BANSAL SB SWAPNIL BANDE Hemangee Kalpesh Kapoor

In theory

- Sequential
  - Time to sum n numbers ?
  - O(n)
  - Time to sort n numbers ?
  - O(n log n)

- Parallel
  - Time to sum n numbers?
  - Tree for O(log n)
  - Time to sort n numbers
  - Non trivially O(log n)

But in practice

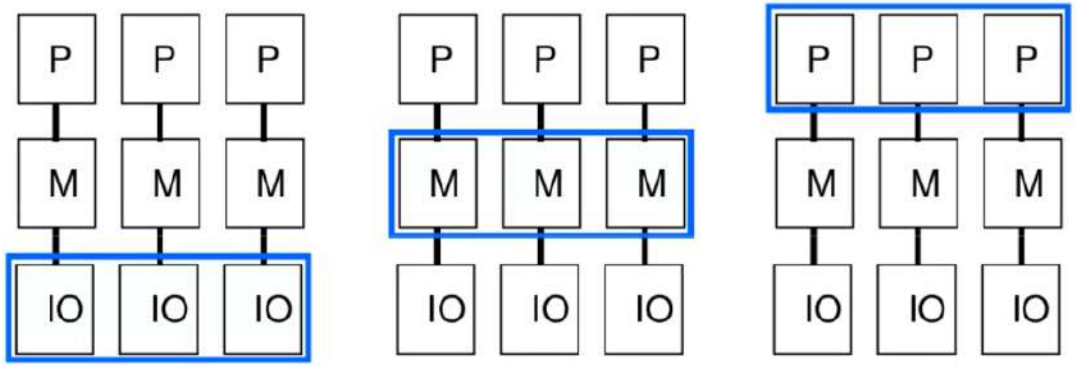
- Name a datum across processors?
- Communicate values?
- Coordinate and synchronize?
- Select processing node size (few-bit ALU to a PC)?
- Select number of nodes in system?

Programming Model

- Provides a communication abstraction that is a contract between hardware and software (a la ISA)
  - Programming model != programming language
- Conceptualisation of the machine that the programmer uses in coding applications
  - How parts cooperate and coordinate their activities
  - Specifies communication and synchronisation operations
- Multiprogramming
  - No communication or synchronisation at program level
- Shared address space
  - Like a bulletin board
- Message Passing
  - Like letters or phone calls, explicit point of contact
- Data Parallel
  - More regimented, global actions on data
  - Implemented using either shared address space or message passing

Historical View

- **Historically: system architecture and programming model were tied together**



Join At:	I/O (Network)	Memory	Processor
Program With:	Message Passing	Shared Memory	Data Parallel Single-Instruction Multiple-Data (SIMD)

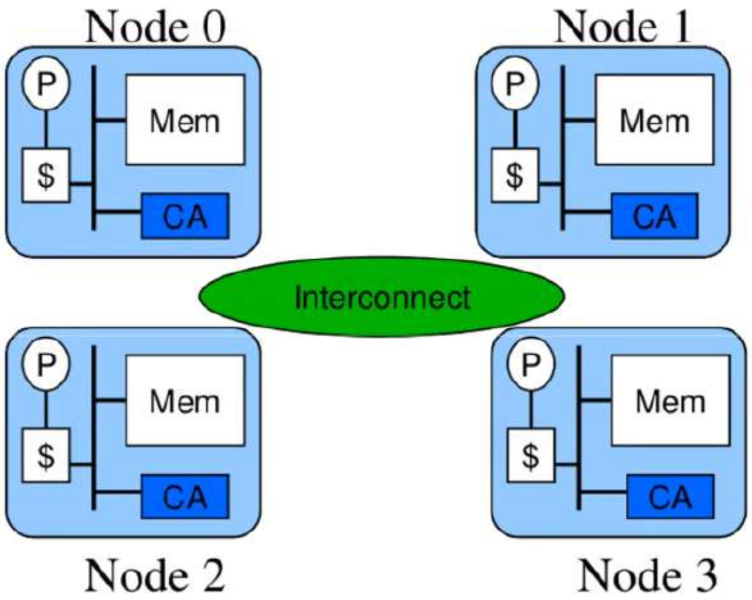
Historical view ...

- Architecture -> Programming Model
  - Join at network -> program with message passing model
  - Join at memory -> program with shared memory model
  - Join at processor -> program with SIMD or data parallel

- Programming Model -> Architecture
  - Message-passing programs on message-passing arch
  - Shared-memory programs on shared-memory arch
  - SIMD/data-parallel programs on SIMD/data-parallel arch

- But
  - Isn't hardware basically the same? Processors, memory, & I/O?
  - **Convergence!** Why not have generic parallel machine & program with model that fits the problem?

A Generic Parallel Machine

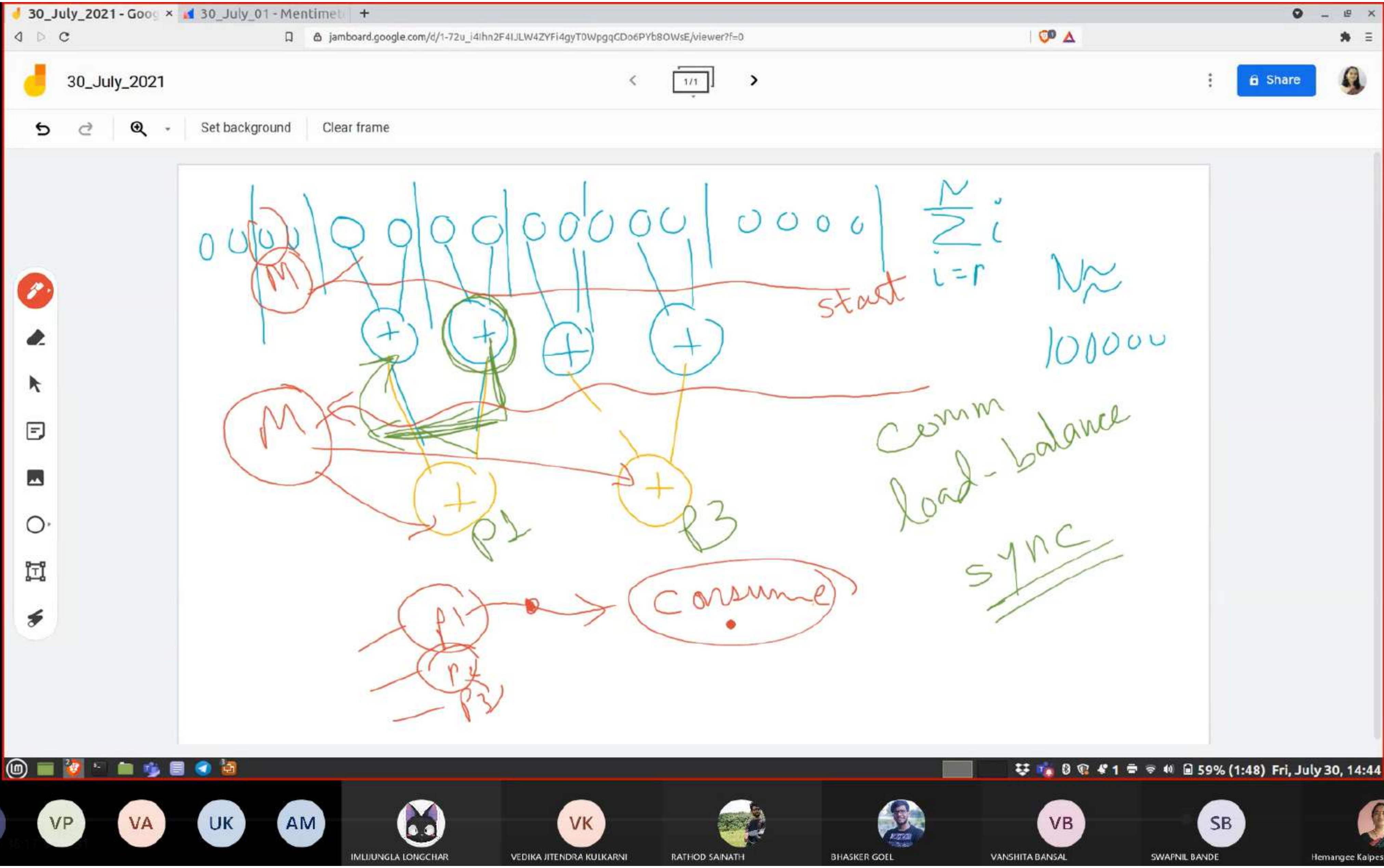


- Separation of programming models from architectures
- All models require communication
- Node with processor(s), memory, communication assist

Reminder: could be on one chip or many

Coordination

- When the cores can work independently, writing a parallel program is much same as writing a serial program
- It gets complex when the cores need to coordinate their work
- Coordination involves
  - Communication
    - Send partial sums to another core
  - Load Balancing
    - We want each core to do same amount of work. Otherwise some cores are working and others are waiting idle wasting computational power
  - Synchronisation
    - Master reads the values in the array
    - Then cores must start computing. Else wait
    - Add a point of synchronisation between read value and compute partial sums


30_July_2021 - Goo
30_July_01 - Mentimet
jamboard.google.com/d/1-72u_i4lhn2F4IJLW4ZYFi4gyT0WpgqCDo6PYb8OWsE/viewer?f=0
30_July_2021
1/1
Share
Set background
Clear frame
start
N Σ i=r i
N ≈
100000
M
P1
P3
comm
load-balance
sync
consume
59% (1:48) Fri, July 30, 14:44
+52
VP
VA
UK
AM
IMLIJUNGLA LONGCHAR
VK
VEDIKA JITENDRA KULKARNI
RATHOD SAINATH
BHASKER GOEL
VB
VANSHITA BANSAL
SB
SWAPNIL BANDE
Hemangee Kalpesh Kapoor

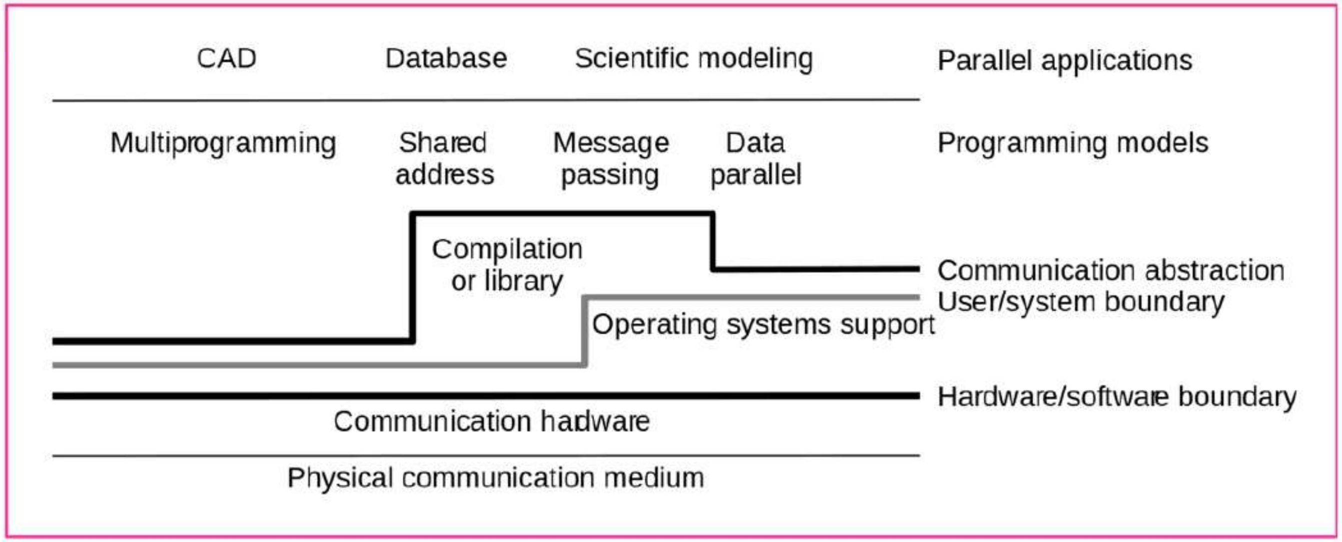
Parallel Programming

- For Distributed Memory and Shared Memory Systems
- Using C and its extensions
- MPI: Message Passing Interface
  - For Distributed memory systems
  - Libraries of type definitions + functions + macros
- POSIX threads: pthreads
  - For shared memory systems
  - Libraries of type definitions + functions + macros
- OpenMP
  - For shared memory systems
  - Library + modifications to C compiler

Parallel Architectures

- Almasi and Gottlieb 1989
  - "Parallel computer is a collection of processing elements that communicate and cooperate to solve large problems fast"

Layers of Abstraction



As programming models have become better understood and Implementation techniques have matured, compilers and run-time libraries have grown to provide an important bridge between the programming model and the underlying hardware

Layers of abstraction in parallel computer architecture

- The framework to understand communication in parallel machines is shown in the figure (another slide)
- Top layer = programming model
  - Specifies how the programs running in parallel communicate information to one another, and
  - What synchronisation operations are available to coordinate activities
- Communication abstraction = user level communication primitives
  - Provided directly by hardware, or
  - By operating system, or
  - Machine specific user software

Layers of abstraction in parallel computer architecture

- The framework to understand communication in parallel machines is shown in the figure (another slide)
- Top layer = programming model
  - Specifies how the programs running in parallel communicate information to one another, and
  - What synchronisation operations are available to coordinate activities
- Communication abstraction = user level communication primitives
  - Provided directly by hardware, or
  - By operating system, or
  - Machine specific user software
- Hardware/Software Boundary = almost flat as most hardware features are uniform in complexity

Shared Address Space, Interconnects, Message Passing

+52 VP VA UK AM

IMLIJUNGLA LONGCHAR VEDIKA JITENDRA KULKARNI RATHOD SAINATH BHASKER GOEL VANSHITA BANSAL SWAPNIL BANDE Hemangee Kalpesh Kapoor