PERFORMANCE ENGINEERING

Lecture 8: Large-scale systems and applications

May 19th, 2022

Ana Lucia Varbanescu

a.l.varbanescu@uva.nl

Quizzes last time(s) – L7Q1

Consider SpMV.

Assume the matrix is 4096 x 4096. You split the matrix over the 16 cores on a DAS node, row-wise. The vector is replicated.

You then transform the matrix in CSR, perform computation, and concatenate results.

- What is the main performance problem?
- What PC’s you need for it?

L7Q1 – Performance counters

- Main problem:
  - **Load imbalance**
  - NUMA problems
  - Cache (false) sharing (?)

- Identify it:
  - **Check counter on instructions (executed/retired)**
  - Check sleep/idle status
  - Check remote data transfers

Quizzes last times – L6Q1

- Please design a statistical model to estimate the performance of a parallel histogram for a single (very large) image.
- Please explain how you will design your model, considering the following questions.
  - What output?
  - What features?
  - Collect training data?
  - Validation? Accuracy?

L6Q1

- Output
  - Execution time / number of cycles for a given image
  - Some form of contention / delay due to false sharing or other issues
- Input (features)
  - dimensions of the image (width and height)
  - distribution of the pixel colors (if atomic operations are used).
  - Image entropy
  - number of threads
  - machine features
- Collect data
  - Images (generated/downloaded)
  - … and their timing
  - Cost?
- Validation
  - Compare to measurement and compute error for unseen images

Today

- What about large-scale systems and applications?
- Models, tools, examples …
- … and demo.

Distributed (memory) systems

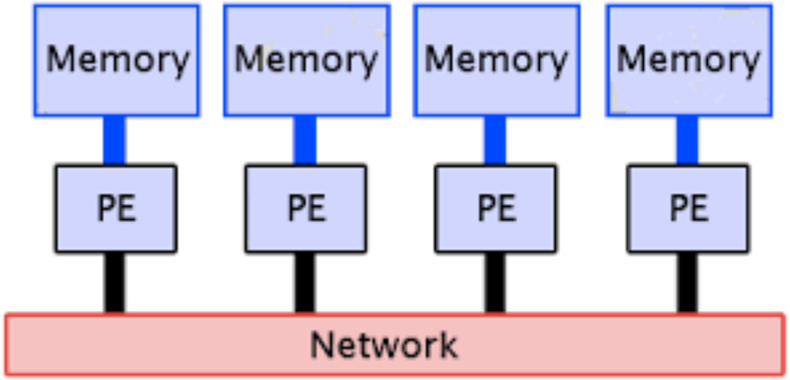
Distributed (memory) systems

- Same principle for performance modeling:

T = F (T_compute, T_comm, T_sync) =

F (f(T_compute_node(i)) , g(T_comm(i)) , h(sync))

F = ? , f,g,h = ?



Distributed (memory) systems

- Same principle for performance modeling:

T = F (T_compute, T_comm, T_sync) =

F (f(T_compute_node(i)) , g(T_comm(i)) , h(sync))

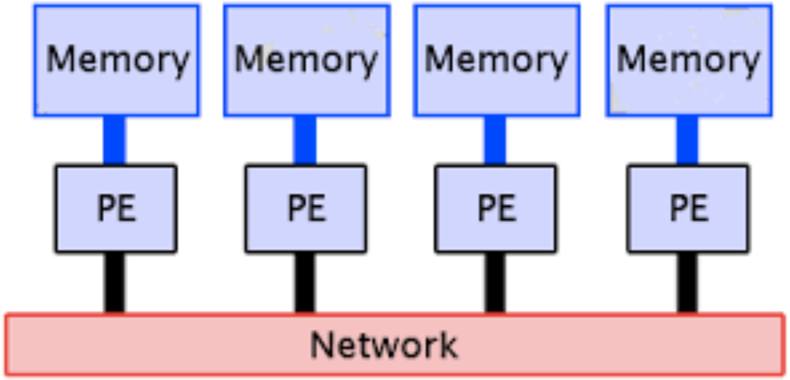
F = + (BSP model)

T = f (T_compute_node(i)) + g() + h()

F = max (overlapping compute and communicate)

T = max (f(), g(), h())

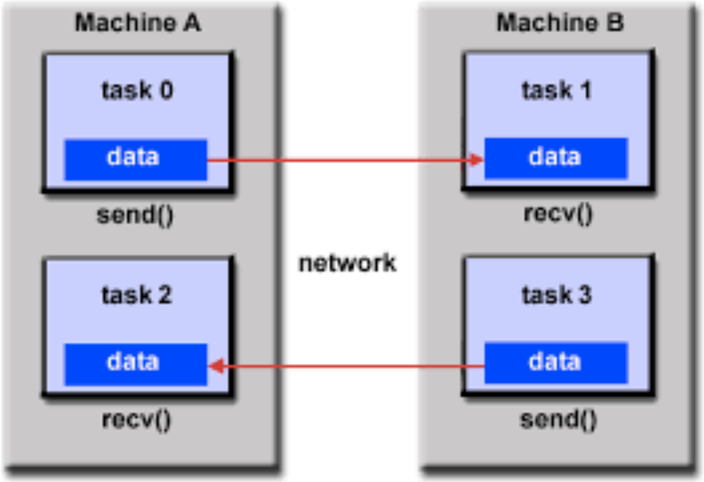
- Boils down to:
  - Understand synchronization model
  - Understand communication model



Typical programming

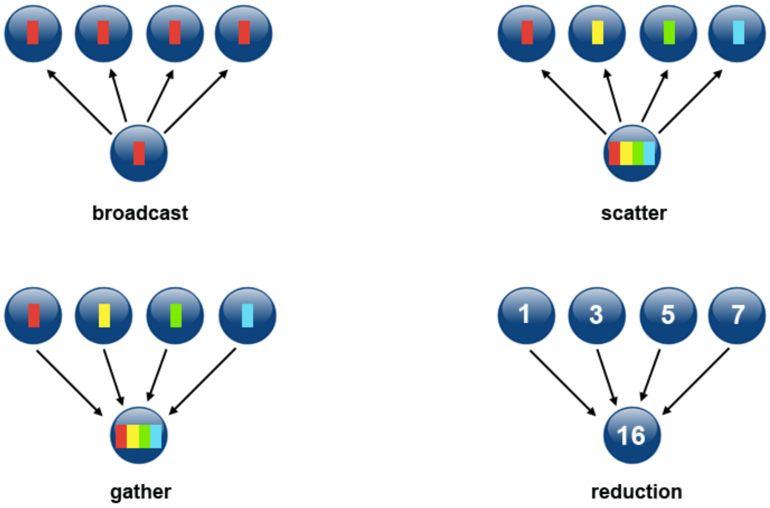
- Message passing (e.g., MPI)

- In message passing, all communication is explicit!
  - Advantage?
  - Disadvantage?

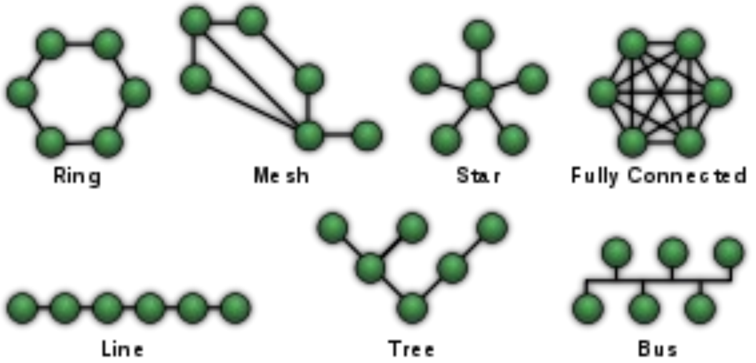


Cost of communication

- Message size
  - The larger, the more time consuming
- Message type
  - Point-to-point
  - Collective communication



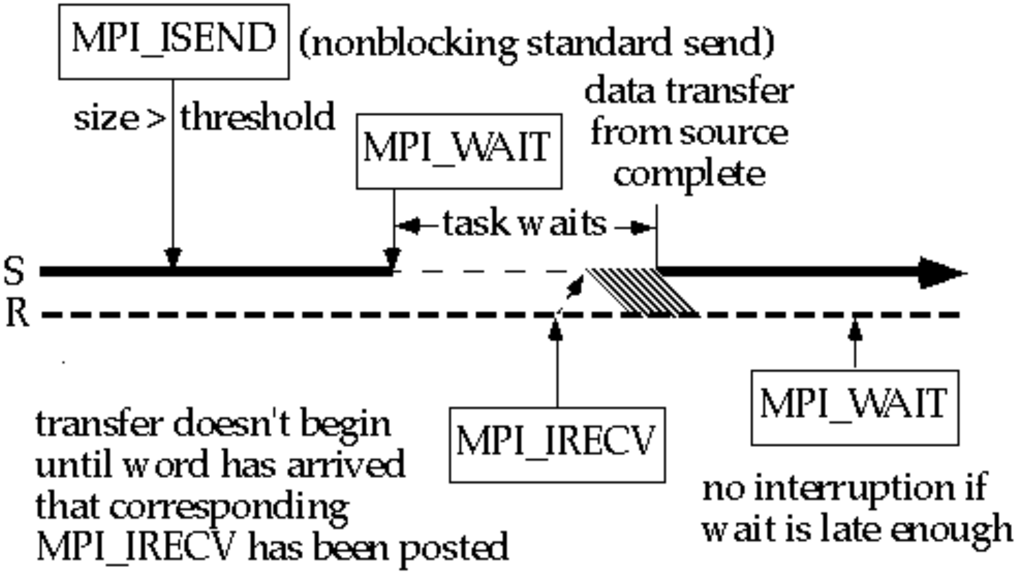
- Network topology



- Not enough knowledge?

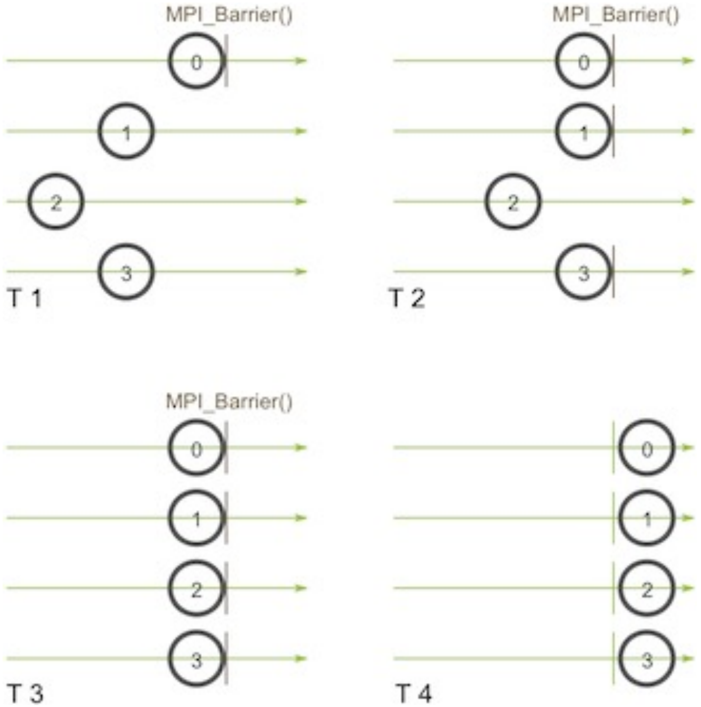
Cost of synchronization

- Blocking vs. non-blocking messages
  - Who is blocking ?
    - Who pays the overhead
  - When is it unblocking ?
  - How much overlap happens?
  - What is the cost of a wait?


MPI_ISEND
(nonblocking standard send)
size > threshold
MPI_WAIT
data transfer from source complete
task waits
S
R
transfer doesn't begin until word has arrived that corresponding MPI_IRECV has been posted
MPI_IRECV
MPI_WAIT
no interruption if wait is late enough

Cost of synchronization

- Barriers
  - Global
  - Per groups



The good news …

- Models focus on scaling behaviour and extrapolation
  - And its potential bottlenecks
- Most current models are a mix of …
  - High-level analytical models
  - Communication patter analysis
  - Heavy calibration
- There is increasing tool support for such applications.
- See presentation by Dr. Alexandru Calotoiu, ETH Zurich
  - Here: https://canvas.uva.nl/files/6728556/download?download_frd=1