Stony Brook University
Computer Science

PROGRESS IMBALANCE IN MULTI-PROCESS PERFORMANCE

Arghya Bhattacharya
Algorithms Lab, Computer Science, Stony Brook University

GRD 2021

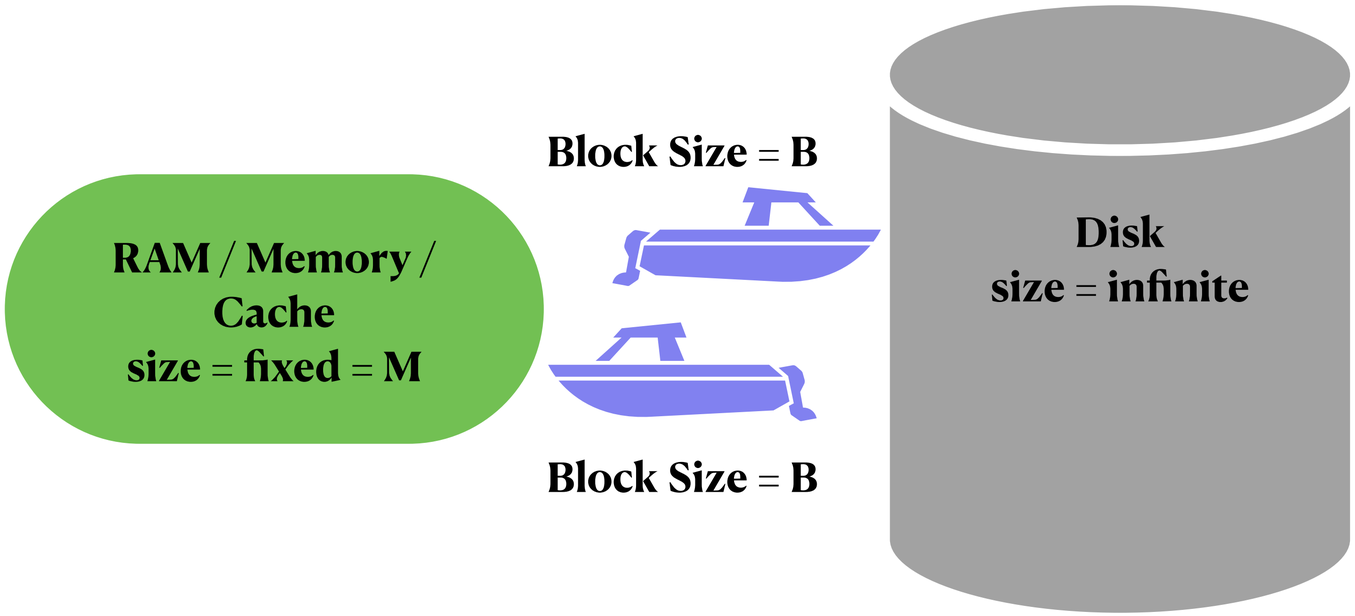
ARCHITECTURE OF MODERN COMPUTERS

- Multi-core
- Multi-threaded
- Time-shared

It is common that multiple programs share a memory

DISK ACCESS MODEL

Stony Brook University
Computer Science



- Two-layered memory hierarchy
- Processing happens in the RAM
- Memory transfer (I/O) is reading from disk to RAM or vice versa
- This happens in chunks of size B.

[Aggarwal and Vitter, Communications of the ACM'88]

A program's running time is dependent on number of I/Os

CACHE SIZE MATTERS

Stony Brook University
Computer Science

Example of I/O complexity:

Strassen's algorithm for Matrix Multiplication = $\mathcal{O}(\frac{N^{\log_2 7}}{B\sqrt{M}})$

External-memory Merge Sort = $\mathcal{O}(\frac{N}{B}\log_{\frac{M}{B}}\frac{N}{B})$

Cache available ⬆
I/O made by a program ⬇

IMPORTANT TO UNDERSTAND THE CACHE BEHAVIOR

Stony Brook University
Computer Science

- DAM model assumes M is fixed.
- How is M shared among programs running concurrently?

Do the concurrent programs share a memory gracefully?

DO THE CONCURRENT PROGRAMS SHARE A MEMORY GRACEFULLY?

Stony Brook University
Computer Science

Experiment: Running homogeneous instances concurrently
Homogeneous instances: copies of the same program with same problem size

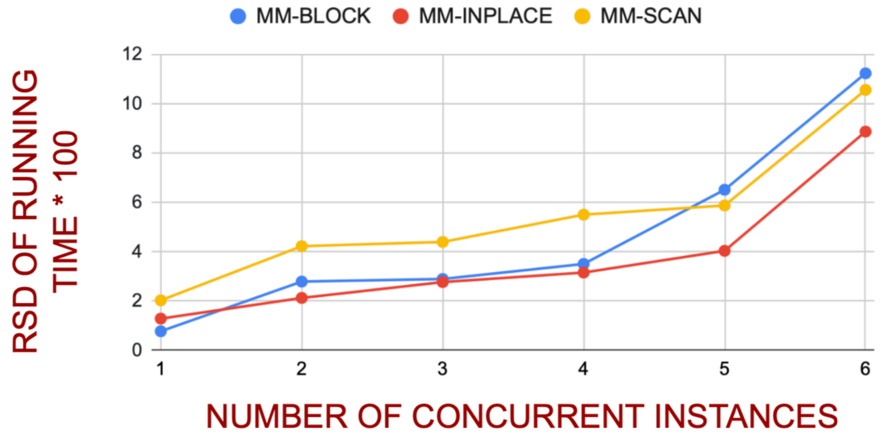
Observation: A difference in the running time.
Progress imbalance: some program instances finish earlier than others.

Number of concurrent processes ⬆
Progress imbalance ⬆

DO THE CONCURRENT PROGRAMS SHARE A MEMORY GRACEFULLY?

Stony Brook University
Computer Science

Experiment: Running a number of homogeneous instances concurrently
Observation: A progress imbalance of the instances



[Bender et al., SPAA'16]

Number of concurrent processes ↑
Progress imbalance ↑

ETIOLOGY OF PROGRESS IMBALANCE

Stony Brook University
Computer Science

Potential reason: Imbalance in cache-sharing

Experiment: We run 6 concurrent instances.
Some instances have a transient stall (let's say x units of time) in the middle.

What do we expect?

- Instances will lose their share of cache when they stall.
- Again gain back their share when they resume.
- They finish with a delay of x units of time.

Is this what we observe?

ETIOLOGY OF PROGRESS IMBALANCE

Stony Brook University
Computer Science

Potential reason: Imbalance in cache-sharing

Experiment: We run 6 concurrent instances.
Some instances have a transient stall (let's say *x* units of time) in the middle.

What do we observe?

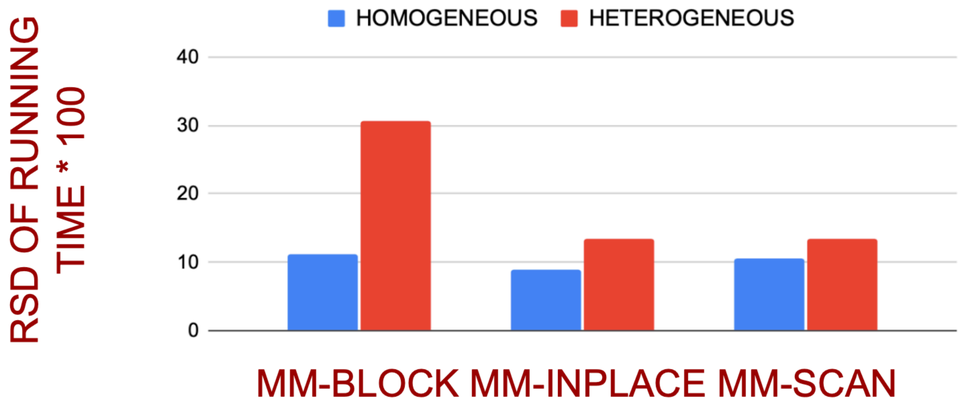
- The instances with a transient stall lose their share of cache.
- They never gain that back.
- They finish much later than the other instances.
- This causes more progress imbalance.

Imbalance in cache-sharing leads to progress imbalance

IMBALANCE IN CACHE-SHARING LEADS TO PROGRESS IMBALANCE

Stony Brook University
Computer Science

Experiment: We run 6 concurrent instances.
Half of the instances have a transient stall (let's say *x* units of time) in the middle.



When multiple homogeneous programs run concurrently and share a cache, there's an imbalance in the cache-sharing, leading to a progress imbalance.

FUTURE DIRECTIONS

Stony Brook University
Computer Science

- Quantify how badly different algorithms are affected by this phenomenon.
- Design algorithms that are susceptive.
- From the experiment, it seems if we randomly stall the programs, they may finish at the same time.

Thanks to:
Prof. Michael A. Bender
Prof. Rezaul A. Chowdhury

Thanks to all of you!