Lecture 25:

Addressing the Memory Wall

(+ One Cool Parallel Algorithm)

Parallel Computer Architecture and Programming
CMU 15-418, Spring 2013

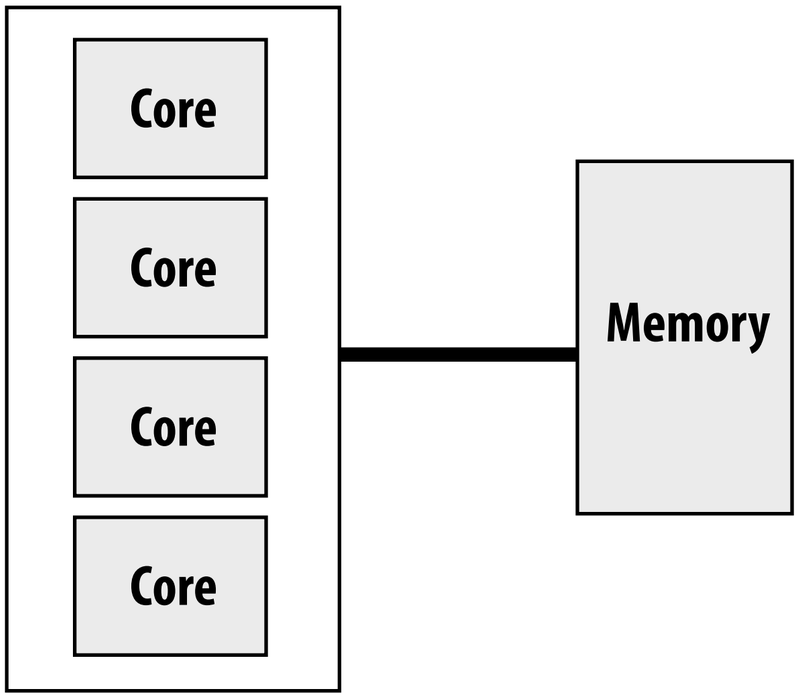
Announcements

- Exam 2 on Tuesday
  - You can bring one post-it note
- Exam review session on Sunday at 5pm
  - Please come with questions
- Final project checkpoint reports due by Friday night

Once again: data transfer is costly!

Limits performance

Multiple processors
= higher aggregate rate of memory requests
= need for more bandwidth
(result: bandwidth-limited execution)



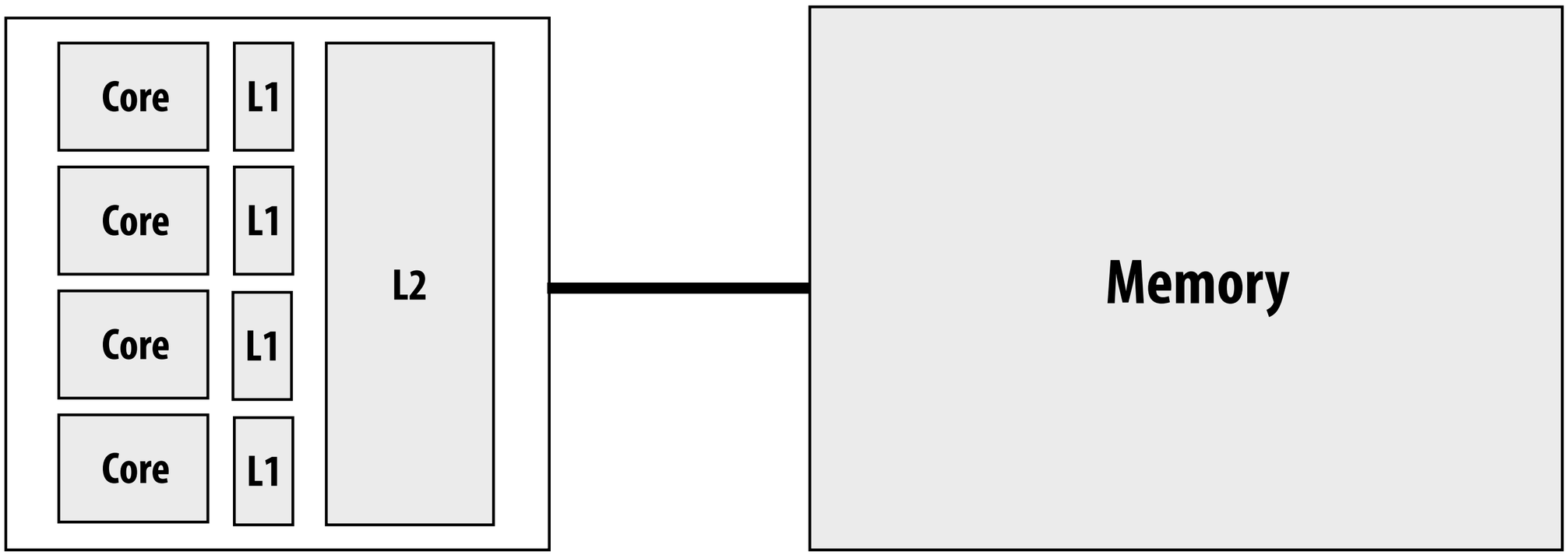
High energy cost

Recall "rough ballpark" numbers from heterogeneity lecture earlier in semester:

~ 20 pJ for an floating-point math op

~1000 pJ to load 64 bits from LPDDR memory

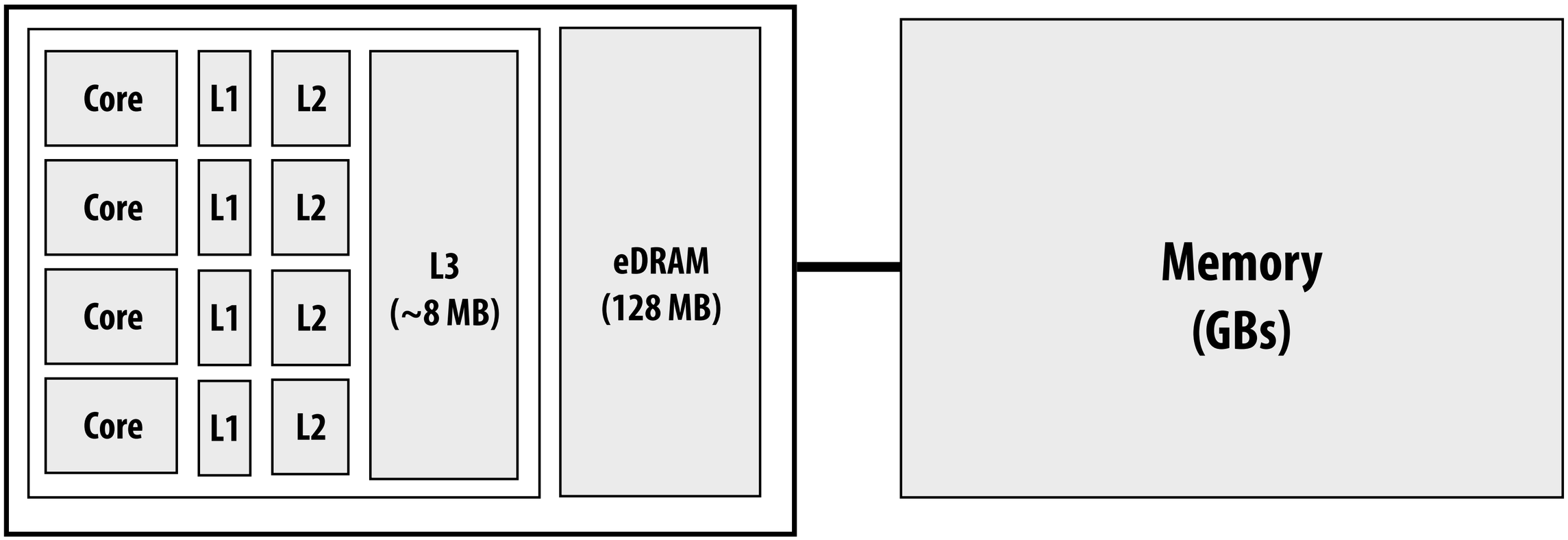
Exploit locality: avoid redundant transfers
(bring frequently used data closer to processor)



- Computations with <u>locality</u> reuse data in local memories (caches, scratchpads)
  - Localize data (load from memory), access many times before discarding/storing it
  - Processor has high-bandwidth (and low latency) access to local memories
    - If computation has locality, most data accessed at this high rate
- Programmers go to great effort to improve the cache locality of their programs

eDRAM: another level of the memory hierarchy

- High-end offerings of the next generation Intel processors (Haswell) will feature 128 MB of embedded DRAM (eDRAM) in the CPU package

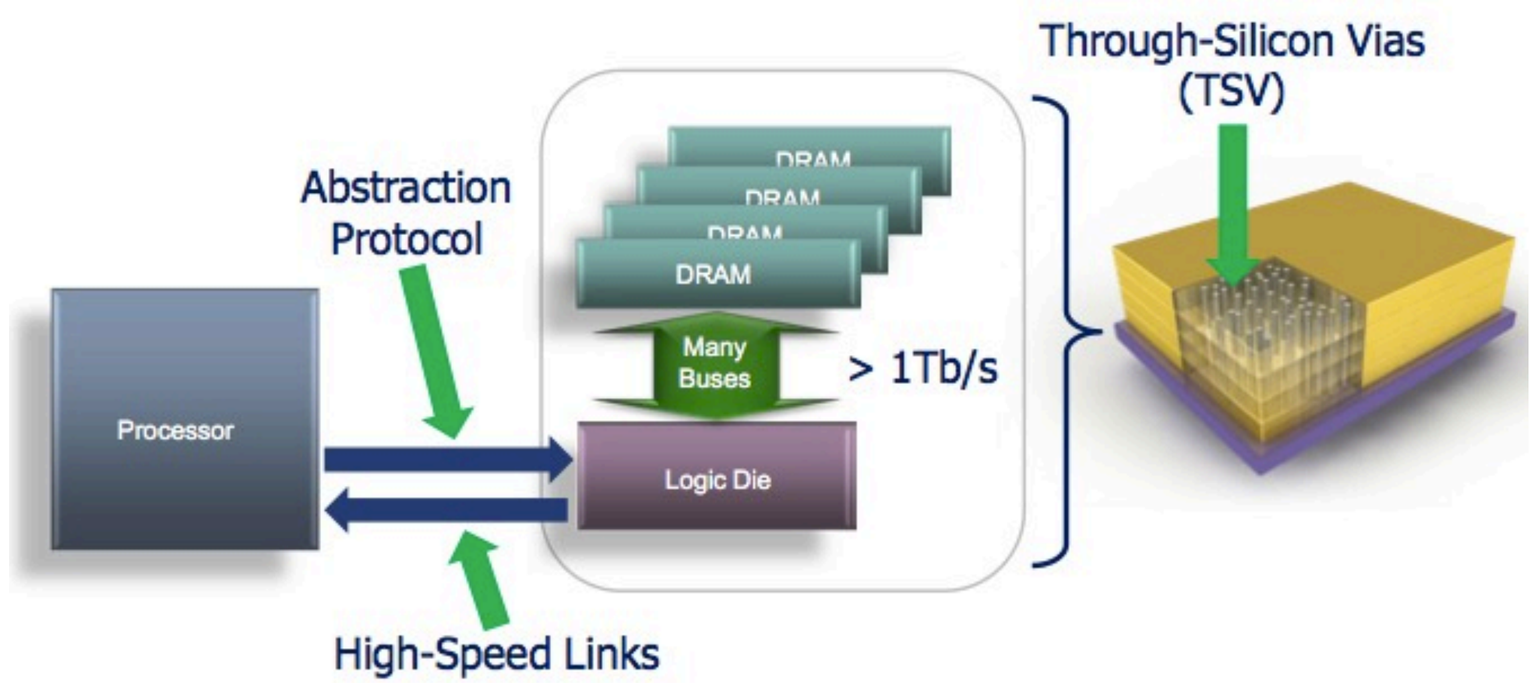


IBM Power 7 server CPUs feature eDRAM

GPU in XBox 360 has 10 MB of embedded DRAM to store the frame buffer

Increase bandwidth: hybrid memory cube

- **Enabling technology: 3D stacking of DRAM chips**
  - DRAMs connected via through-silicon-vias (TSVs) that run through the chips
  - Base layer "logic layer" is DRAM controller, manages requests from processor
  - TSVs provide highly parallel (high BW) connection between logic layer and DRAMs
  - 8-link configuration: 320 GB/sec between CPU and memory cube



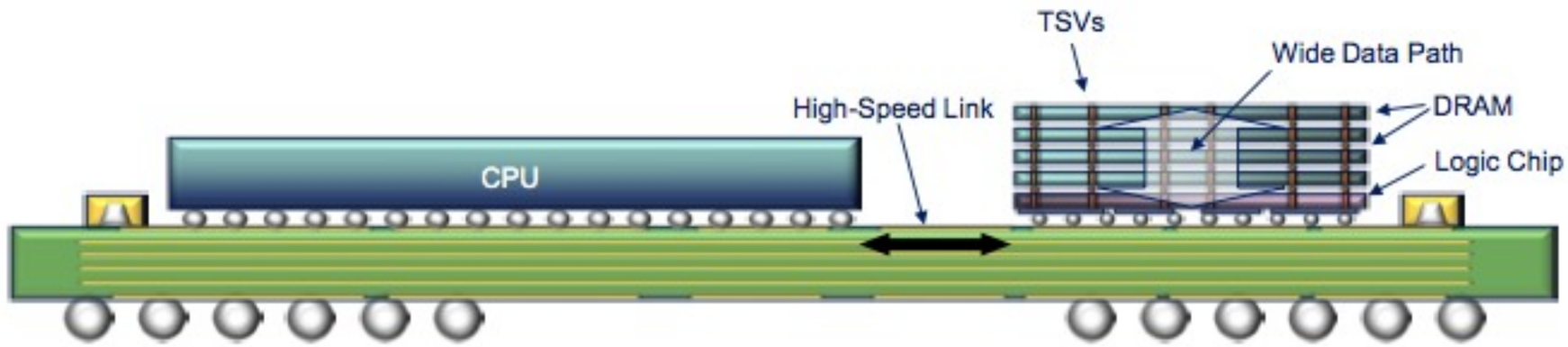


Image credits: Micron, Inc.

Note: height not to scale (actual package not much thicker than a traditional chip)

Move computation to data

Simple example: web application makes SQL query against large user database

Laptop

Web Application Server

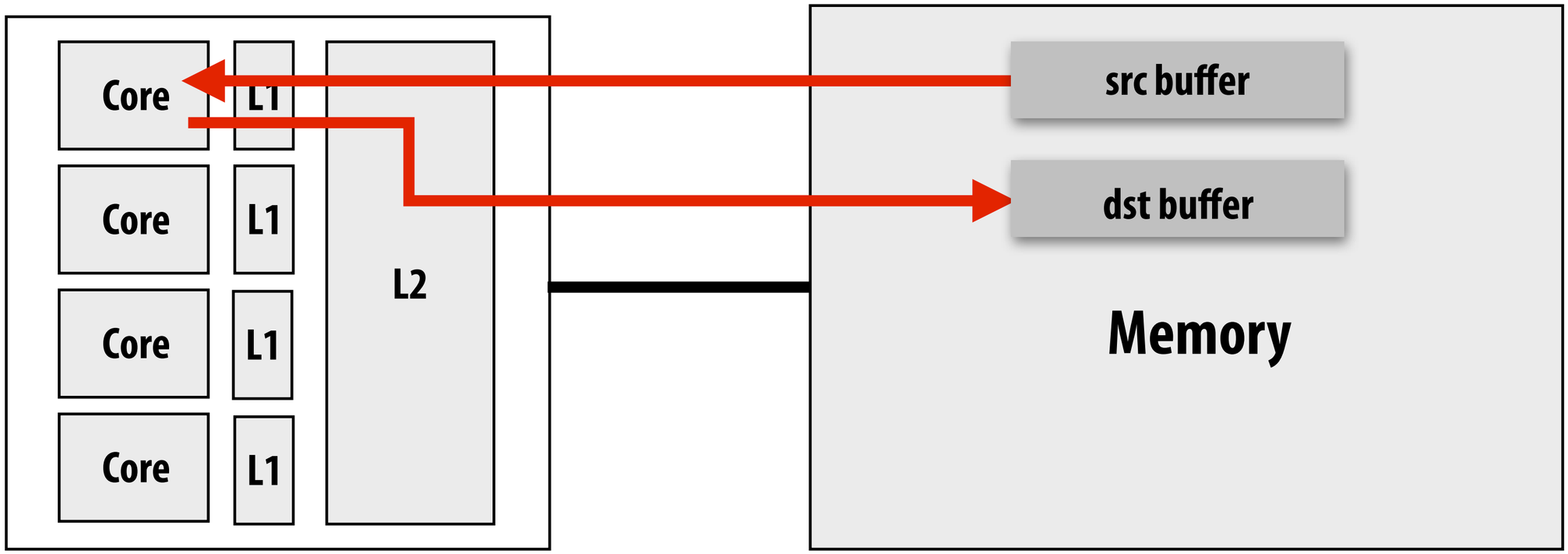
DB Server

DB Server

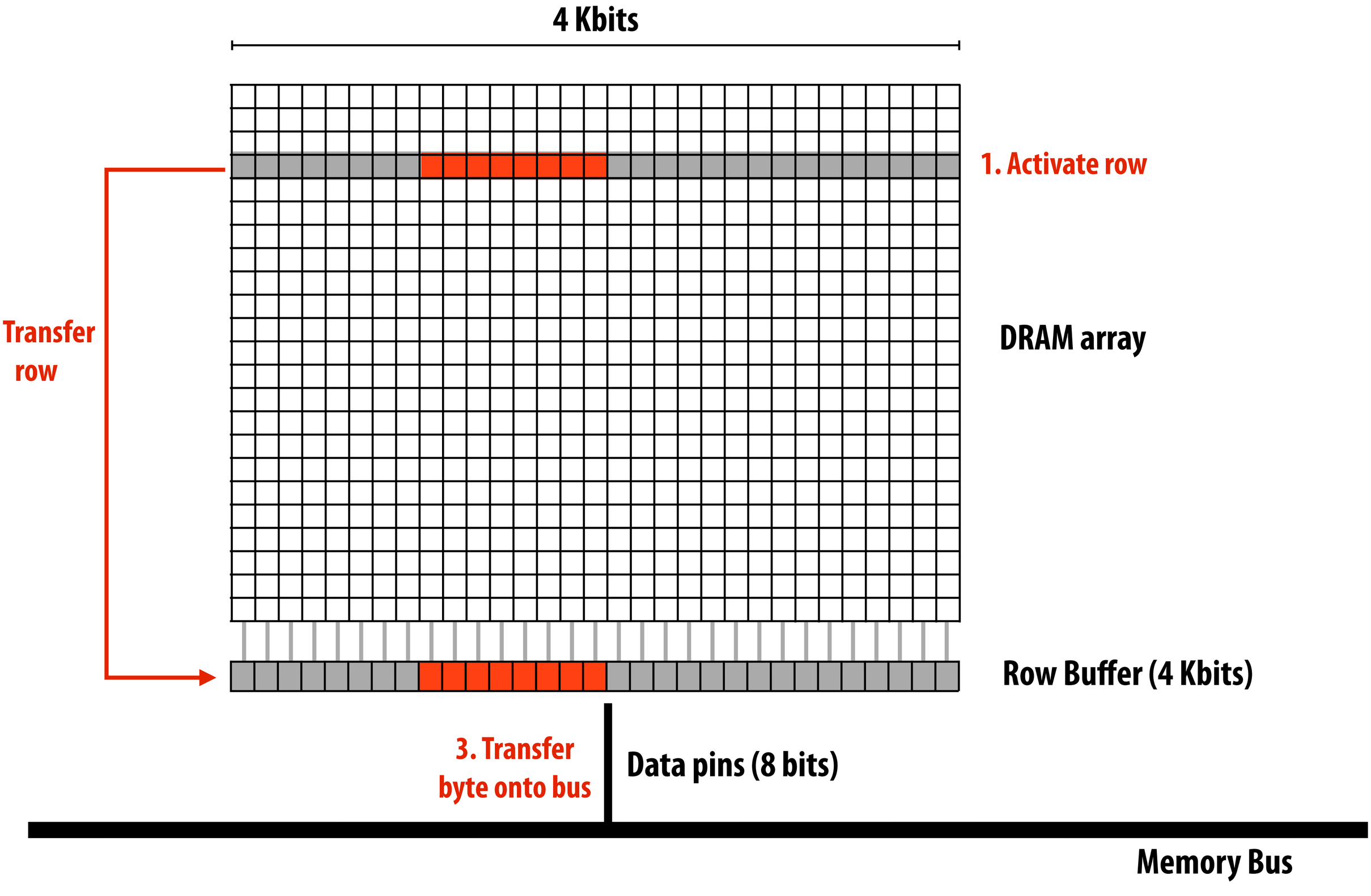
DB Server

DB Server

Memcpy: data movement through entire processor cache hierarchy

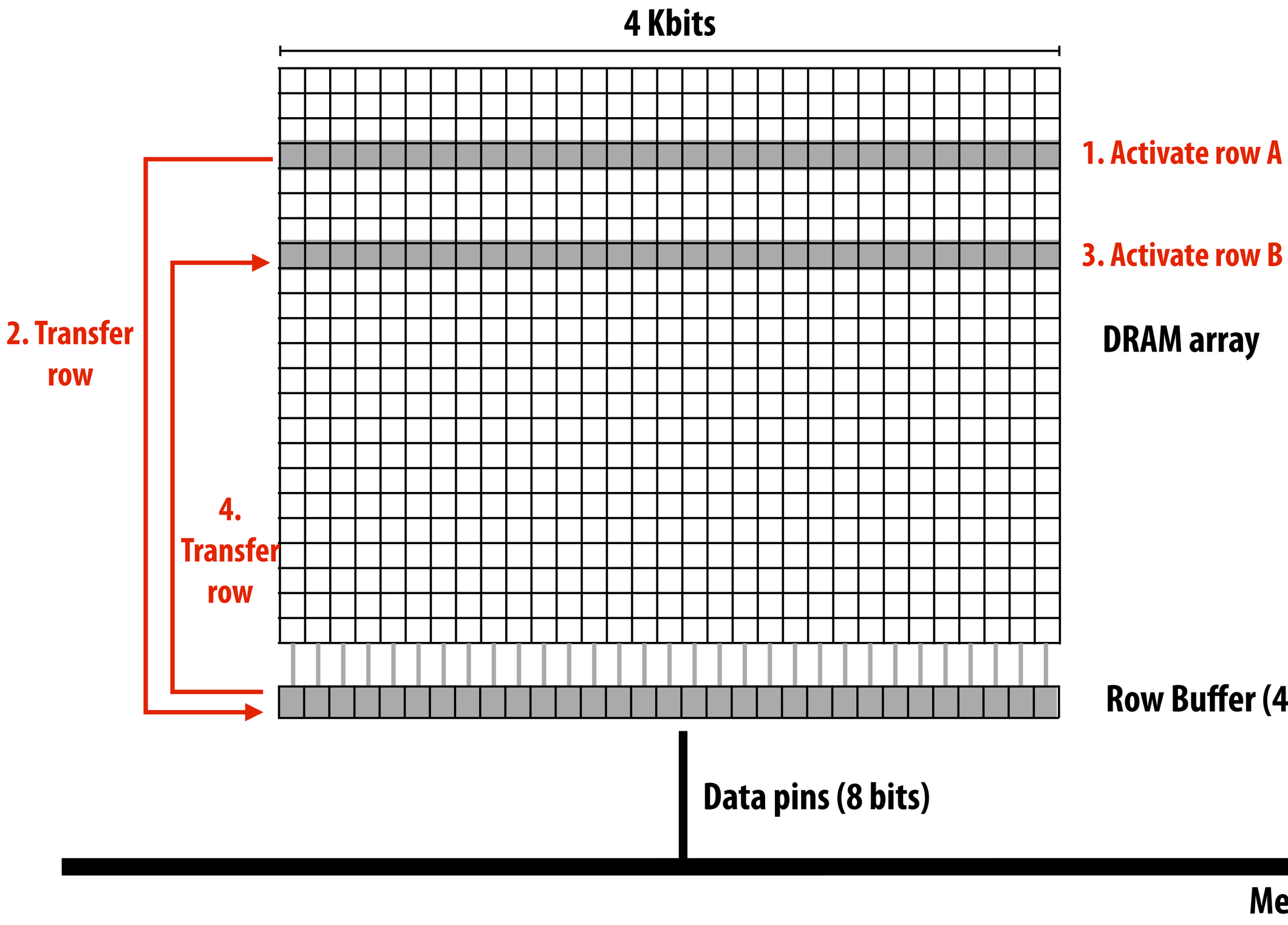


DRAM operation (load one byte)



Idea: perform copy without processor [Seshadri 13]

Modify memory system to support load, store, BULK COPY



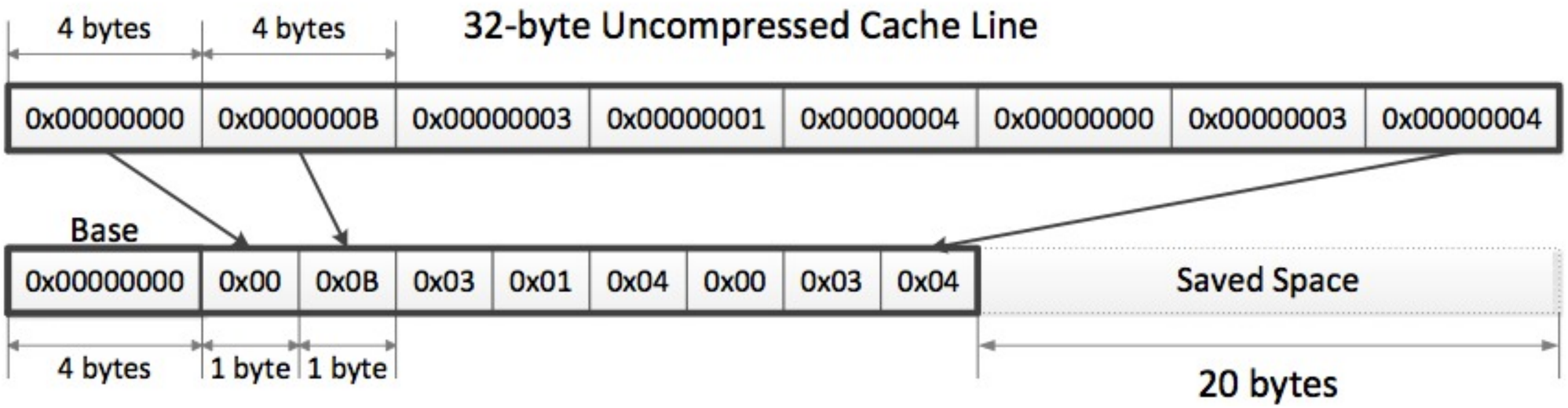
Compress data

- Idea: Increase effective cache capacity by compressing data resident in cache
  - More cache hits = fewer transfers
  - Expend computation (compression/decompression) to save bandwidth

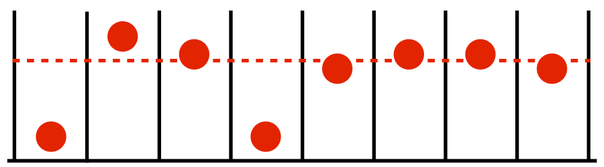
- Compress/decompression scheme must
  - Be simple enough to implement in HW
  - Be fast: decompression is on critical path of load

One proposed example: BΔI compression [Pekhimenko 12]

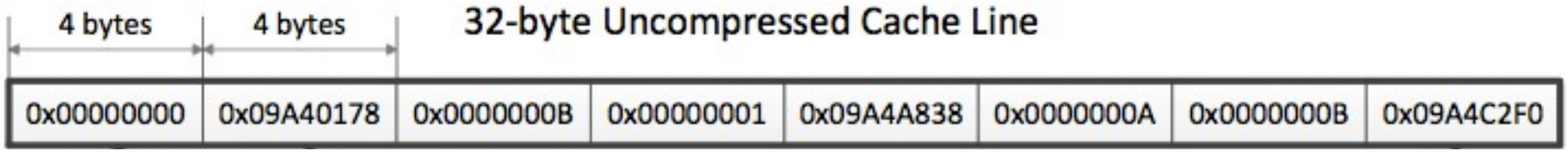
- Observation: data that falls within cache line often has low dynamic range (use base + offset to encode chunks of bits in a line)



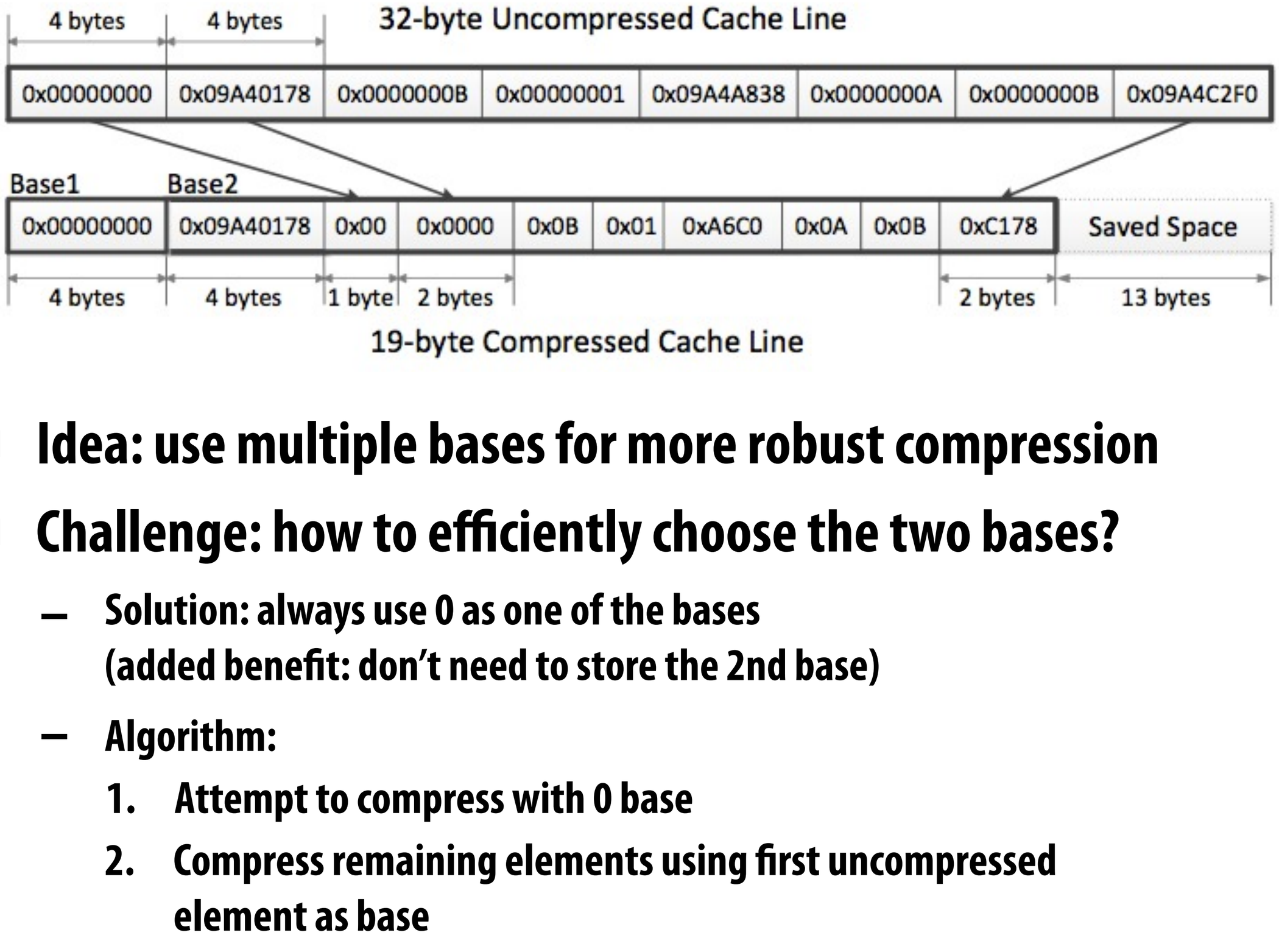
- How does implementation quickly find a good base?
  - Use first word in line
  - Compression/decompression of line is data-parallel



Does this pattern compress well?



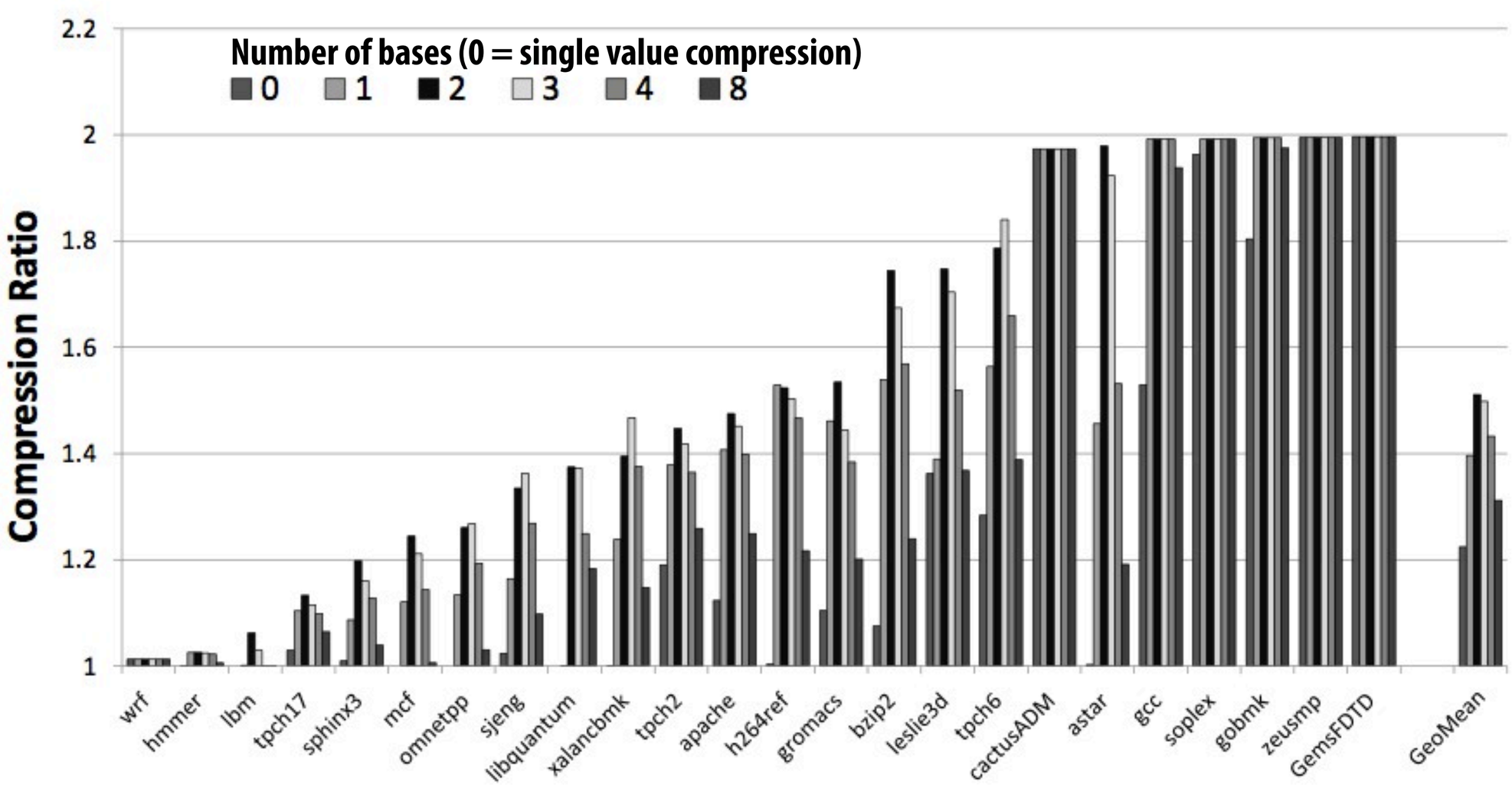
Does this pattern compress well?



- Idea: use multiple bases for more robust compression
- Challenge: how to efficiently choose the two bases?
  - Solution: always use 0 as one of the bases
    (added benefit: don't need to store the 2nd base)
  - Algorithm:
    1. Attempt to compress with 0 base
    2. Compress remaining elements using first uncompressed element as base

Effect of cache compression

[Pekhimenko 12]



- On average: ~ 1.5x compression ratio
- Translates into ~ 10% performance gain, up to 18% on cache sensitive workloads

Bandwidth reducing trick in ARM GPUs

- Frame-buffer write during rendering is a bandwidth-heavy operation
- Idea: skip frame-buffer write if it is unnecessary
  - Frame 1:
    - Render frame tile at a time
    - Compute hash for each tile on screen
  - Frame 2:
    - Render frame tile at a time
    - Before writing pixel values for tile, compute hash and see if tile is the same as last frame
      - If yes, skip write

Slow camera motion: 96% of writes avoided
Fast camera motion: Still ~50% of writes avoided

- GPUs compress frame-buffer contents prior to write to save bandwidth (data compressed in memory, unlike previous example where data was compressed when in cache)

[Source: Tom Olson http://blogs.arm.com/multimedia/780-how-low-can-you-go-building-low-power-low-bandwidth-arm-mali-gpus/]

Summary: memory wall is being addressed in many ways

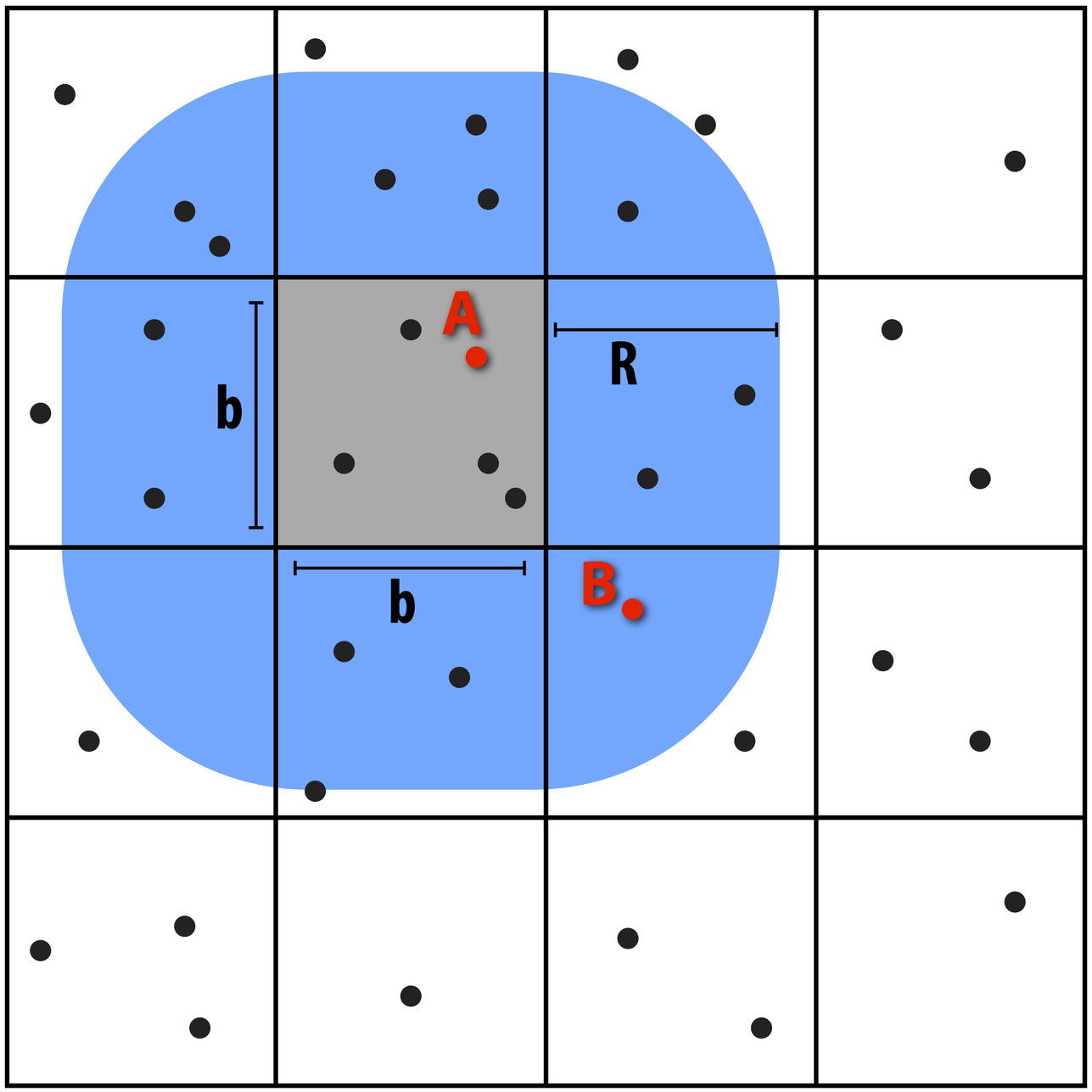
- By application programmer
  - Schedule computation to maximize locality (minimize data movement)
- In hardware implementation by architects
  - Bring data closer to processor (deep cache hierarchies, eDRAM)
  - Increase bandwidth (wider memory systems, near future: 3D stacking)
  - Ongoing research in located limited computation "in" or near memory
- General principles
  - Locate data near processor
  - Move computation to storage
  - Data compression (trade-off extra computation for less data transfer)

Bonus slides:

A fun parallel algorithm that I think you will enjoy: the NT method

Limited range force computation

Goal: compute interactions between all particles located within distance R



Partition space into P regions (one region per processor)

Two sets of particles:

"Home" region: ~ b^2
(particles in gray box, "owned" by processor P)

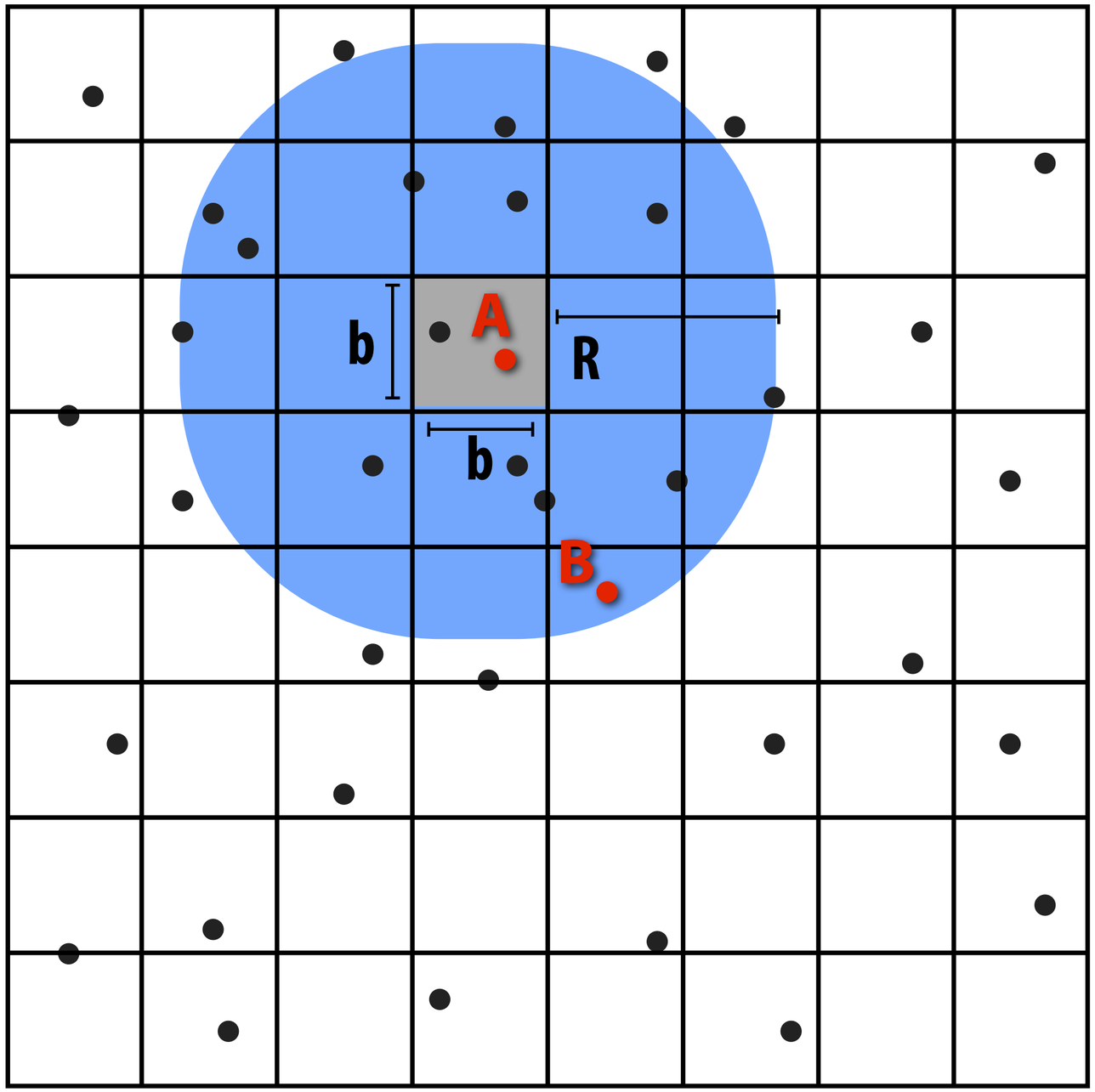
"Import" region: ~ $4bR + \pi R^2$
(particles that must be communicated to processor P to perform computation)

Number of interactions carried out by processor P is proportional to product of the two terms:

~ $b^2(4bR + \pi R^2)$

Limited range force computation: scaling

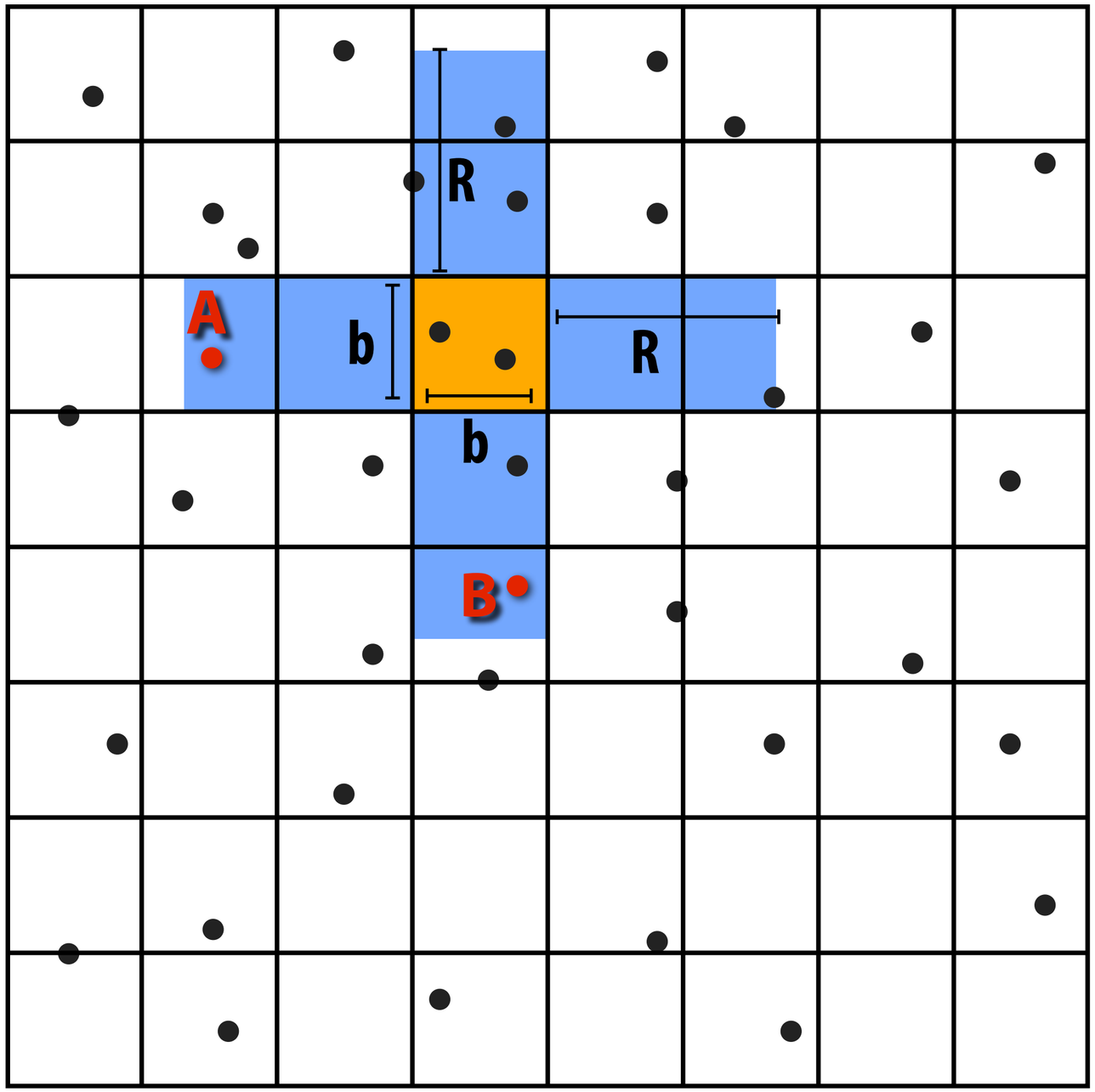
Goal: compute interactions between all particles located within distance R



Consider comm. to comp. ratio as P → ∞:
Note: b ~ 1/sqrt(P)

Import region shrinks to πR^2
Home region shrinks to 0

“Neutral territory” (NT) approach (in 2D)



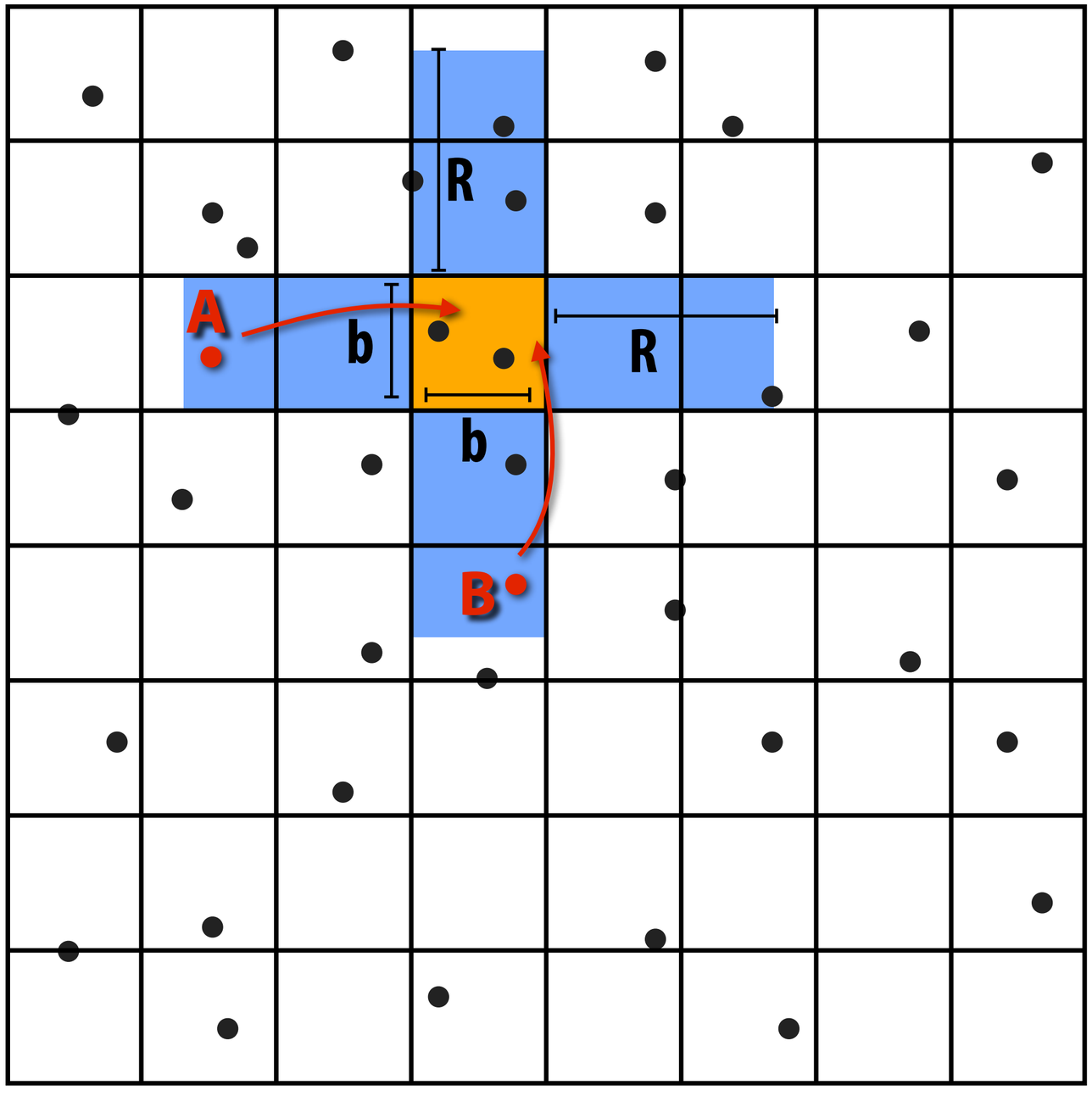
Particle A interacts with particle B in square with x coord equal to that of A and y coord equal to that of B. (yellow square)

Import region of the yellow square associated with processor P is highlighted ROW and highlighted COLUMN.

Size of import region = 4bR

Size of import region as $P \rightarrow \infty = 0$

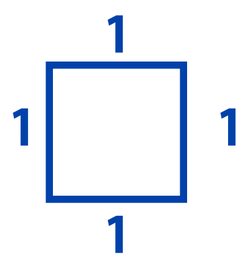
“Neutral territory” intuition



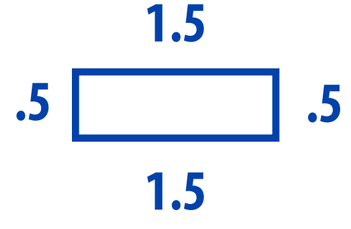
Assuming R > b (true for high processor count), most particle interactions computed by processor upon which neither particle resides!

Intuition: In the NT method the two sets of interacting particles (column and row) are the same size. Recall number of interactions is the <u>product</u> of these sizes.

Analogy: perimeter of a square is greater than any other quadrilateral with the same area.

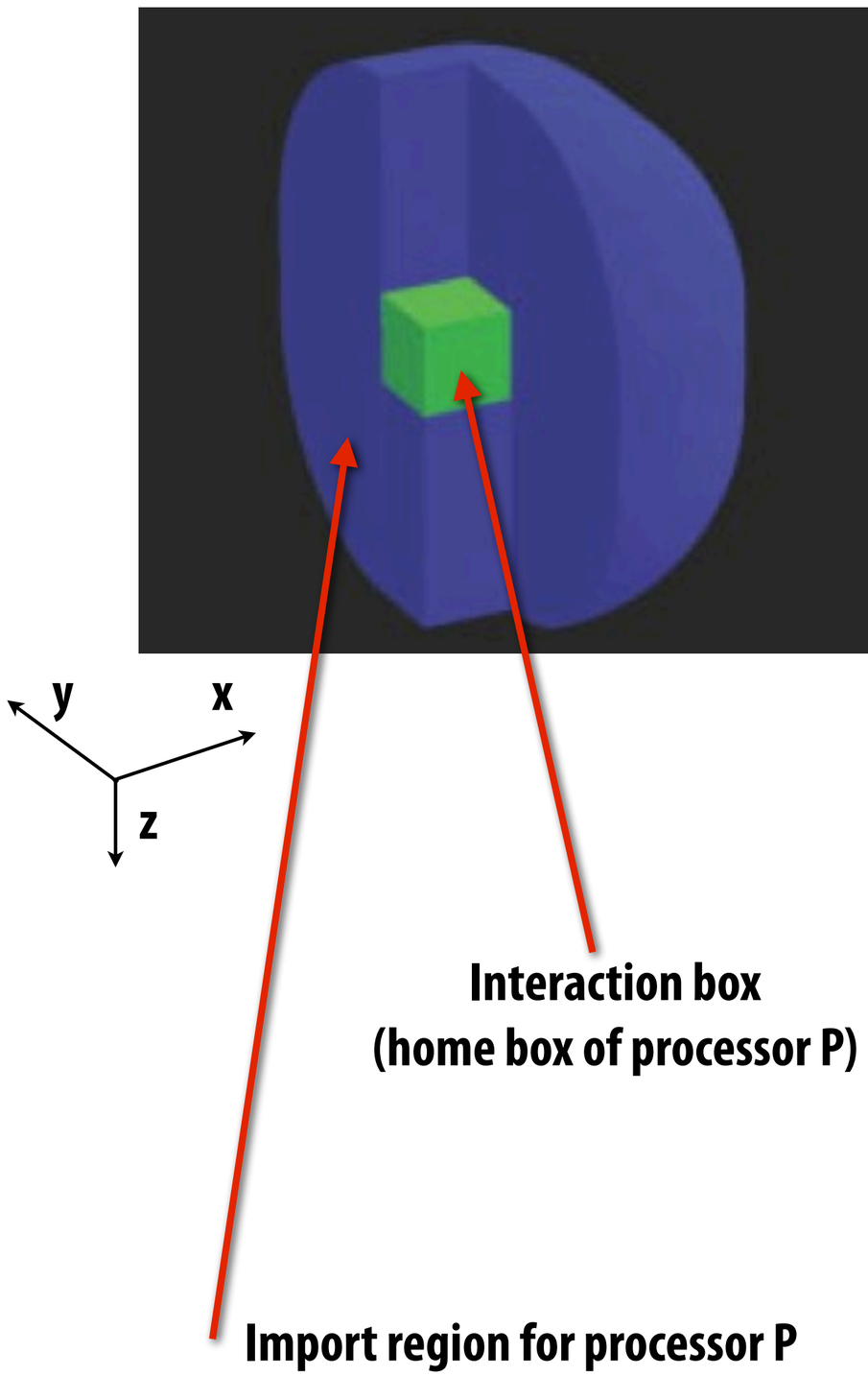


Perim = 4
Area = 1



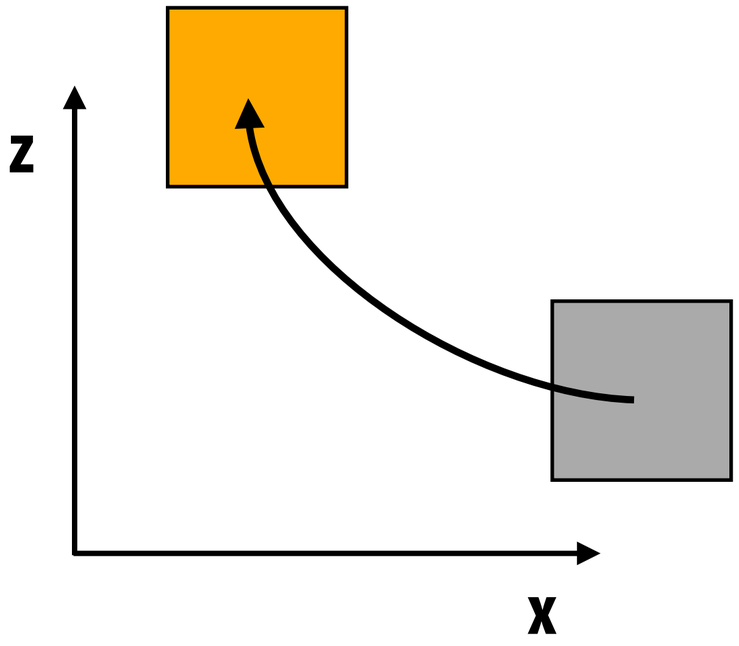
Perim = 4
Area = .75

Extending to 3D



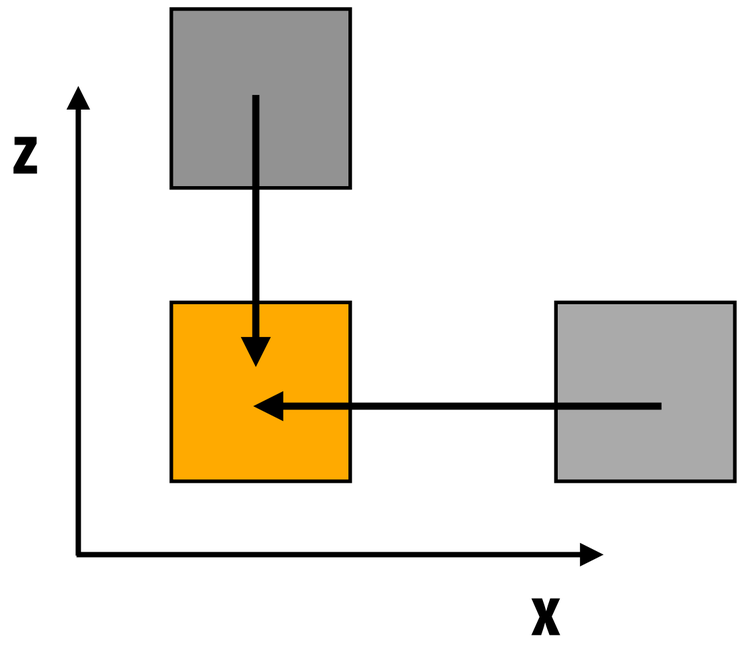
- Divide space into $b_x \times b_y \times b_z$ cells
- Want to avoid computing interaction between each particle pair twice
- Half-shell interaction rules for particles A and B (applied in this order):
  - If $A_x < B_x$ interact in home box of A
  - If $A_y < B_y$ interact in home box of A
  - If $A_z < B_z$ interact in home box of A
  - If A and B are in the same box, no movement is necessary for interaction.

NT intuition



Half-Shell Method

Perform computation in box with smaller X coordinate

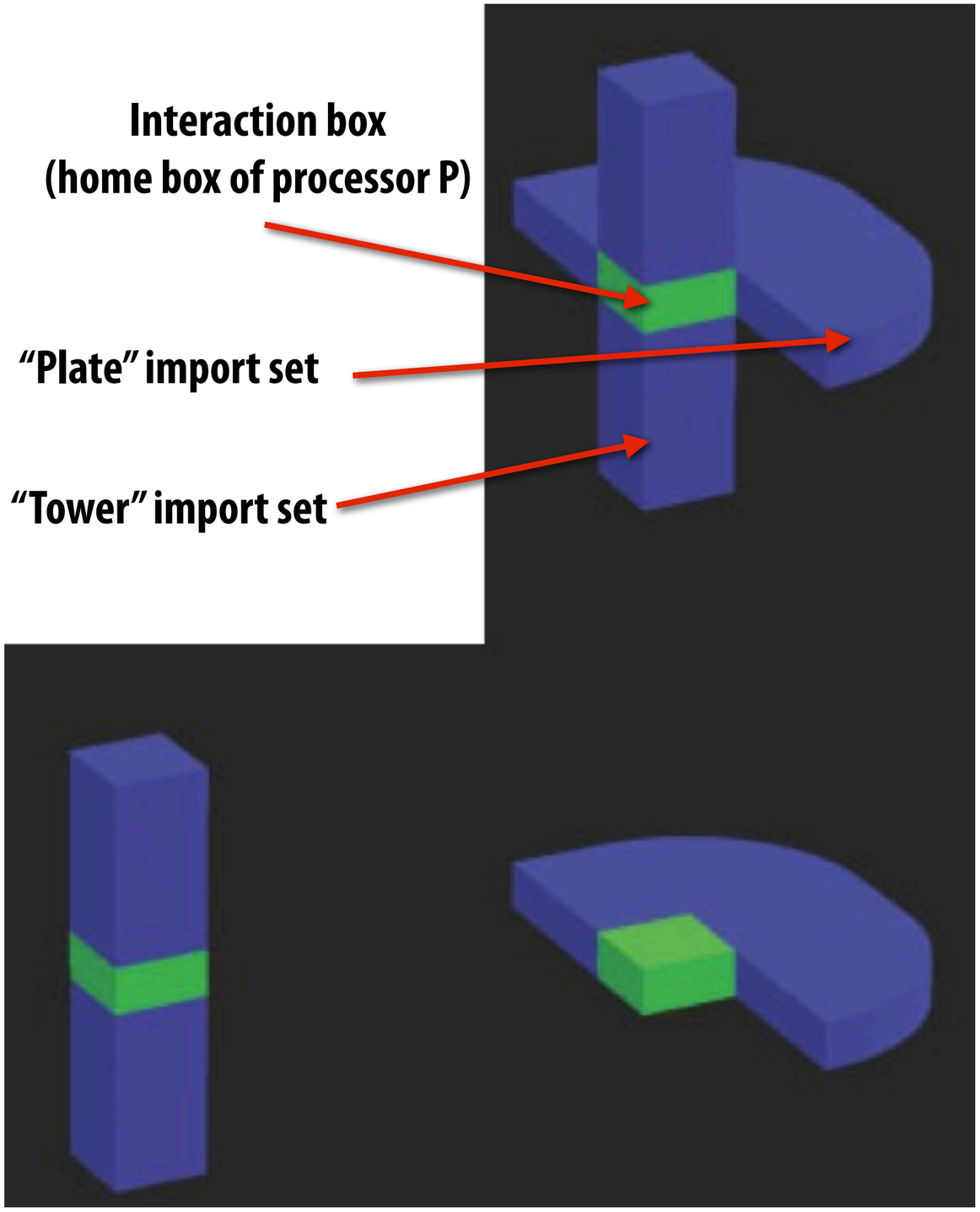


Neutral-Territory Method

Perform computation in box with X coordinate from particle with smaller X coordinate and Z coordinate from particle with larger X coordinate

= computation performed by processor responsible for this region of space

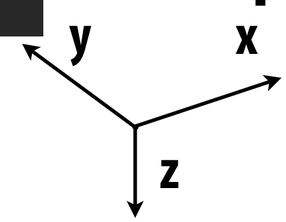
Extending NT method to 3D



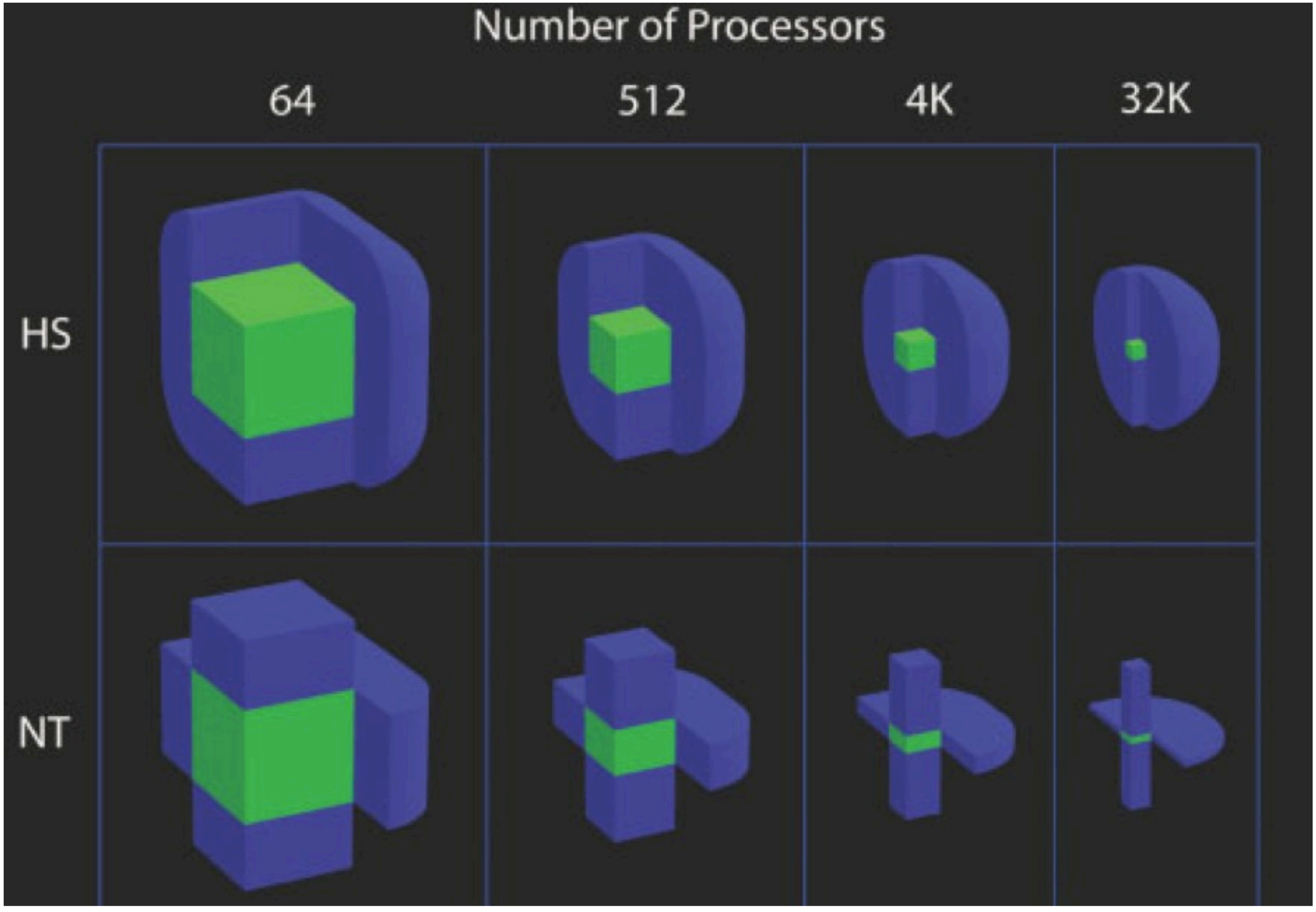
- Divide space into $b_x \times b_y \times b_z$ cells

- NT interaction rules for particles A and B (applied in this order):
  - If $A_x < B_x$ A is in the tower
  - If $A_y < B_y$ A is in the tower
  - If $A_z < B_z$ A is in the plate
  - If A and B are in the same box, choose arbitrarily

- Two particles will interact in box with x,y coordinate of tower and z coordinate of plate



Scaling



NT method asymptotics (3D)

- **Half-shell method**
  - import volume $V = 3Rb^2 + (3/2)\pi R^2 b + (2/3)\pi R^2$
  - As $P \rightarrow \infty$: $V = (2/3)\pi R^2$

- **NT-method**
  - import volume $V = 2Rb_{xy}^2 + 2Rb_z^2 + (1/2)\pi R^2 b_z$
  - As $P \rightarrow \infty$: $V = O(R^{3/2} / p^{1/2})$

Note: b ~ cube root of p

NT method summary

- In N-body problems, communication-to-computation ratio increases as number of processors gets large

- Unintuitive solution: reduce communication requirements by ALWAYS communicating
  - Pick a "neutral processor" to perform computation between two particles