ISC 2024 IXPUG Workshop

Performance Evaluation and Optimization of Seismic Imaging Applications on HBM-Enabled CPUs

Huda Ibeid, Pavel Plotnitskii, Kadir Akbudak, Hatem Ltaief, and David Keyes

Outline

- Stencil computations
  - Spatial blocking
  - Temporal blocking
- Memory Hierarchy and Flow
  - Cache coherence
- Performance results

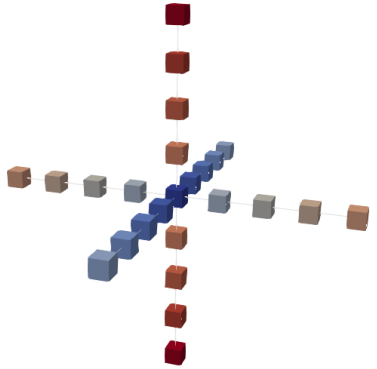
3D Acoustic Wave Equation

- Second-order:

$$\frac{1}{V_P^2(\mathbf{x})}\frac{\partial^2 P(\mathbf{x},t)}{\partial t^2} = \frac{\partial^2 P(\mathbf{x},t)}{\partial x^2} + \frac{\partial^2 P(\mathbf{x},t)}{\partial y^2} + \frac{\partial^2 P(\mathbf{x},t)}{\partial z^2} + \delta(\mathbf{x}-\mathbf{x_s})s(t),$$

- First-order:

$$\begin{aligned}
\frac{1}{V_P^2(\mathbf{x})}\frac{\partial P(\mathbf{x},t)}{\partial t} &= \frac{\partial v_x(\mathbf{x},t)}{\partial x} + \frac{\partial v_y(\mathbf{x},t)}{\partial y} + \frac{\partial v_z(\mathbf{x},t)}{\partial z} + \delta(\mathbf{x}-\mathbf{x_s})s(t) \\
\frac{\partial v_x(\mathbf{x},t)}{\partial t} &= \frac{\partial P(\mathbf{x},t)}{\partial x} \\
\frac{\partial v_y(\mathbf{x},t)}{\partial t} &= \frac{\partial P(\mathbf{x},t)}{\partial y} \\
\frac{\partial v_z(\mathbf{x},t)}{\partial t} &= \frac{\partial P(\mathbf{x},t)}{\partial z},
\end{aligned}$$

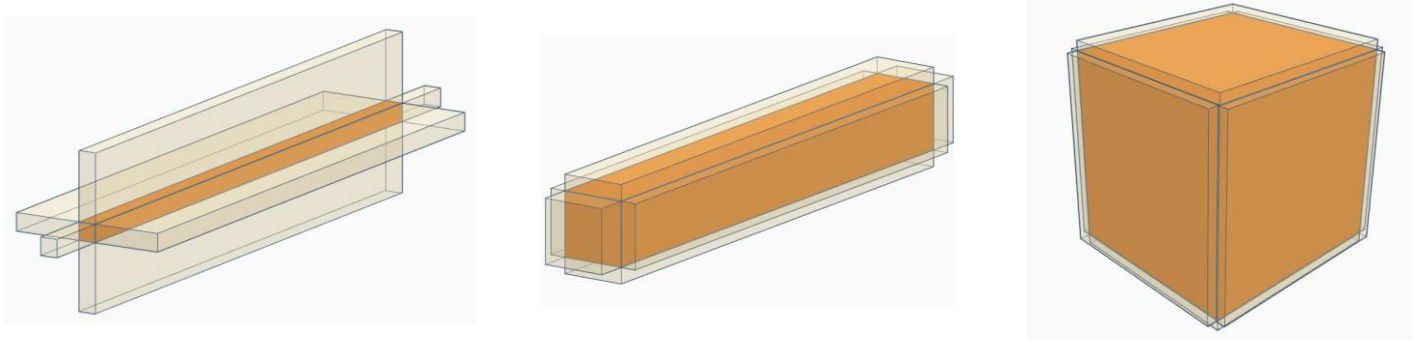


3D finite-difference stencil with 8th order in space with constant coefficients

GIRIH, https://github.com/ecrc/girih

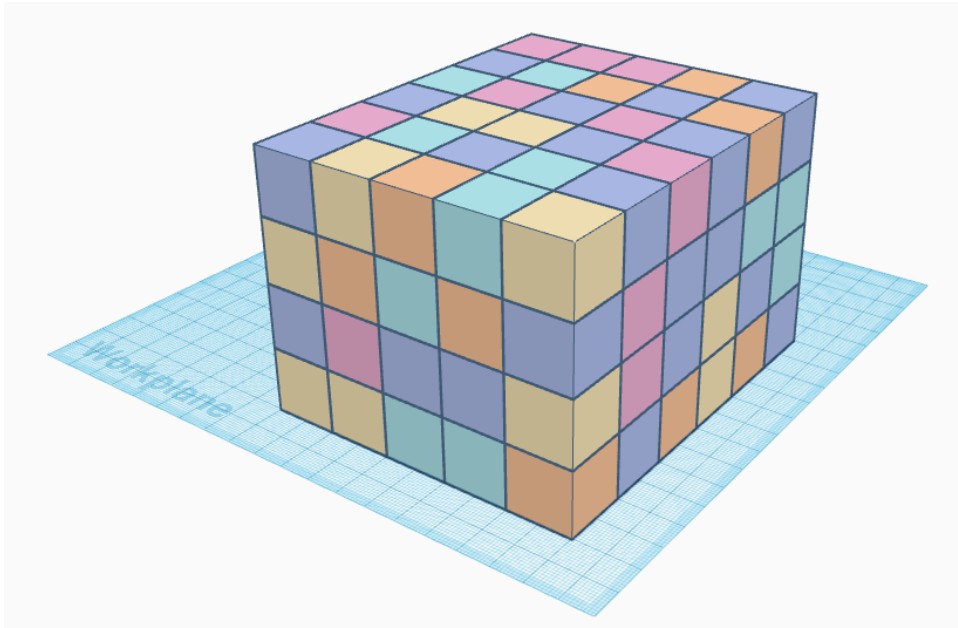
Spatial Blocking (SB)

- Leverage cache reuse.
- Support most of stencil-based applications.
- Provide simplicity and flexibility.



1D 2D 3D

Illustration of Spatial Blocking applied to the stencil computation



3D domain tiled with 3D cube blocks and computed by 6 OpenMP threads (each thread uses a different color)

MWD-TB background

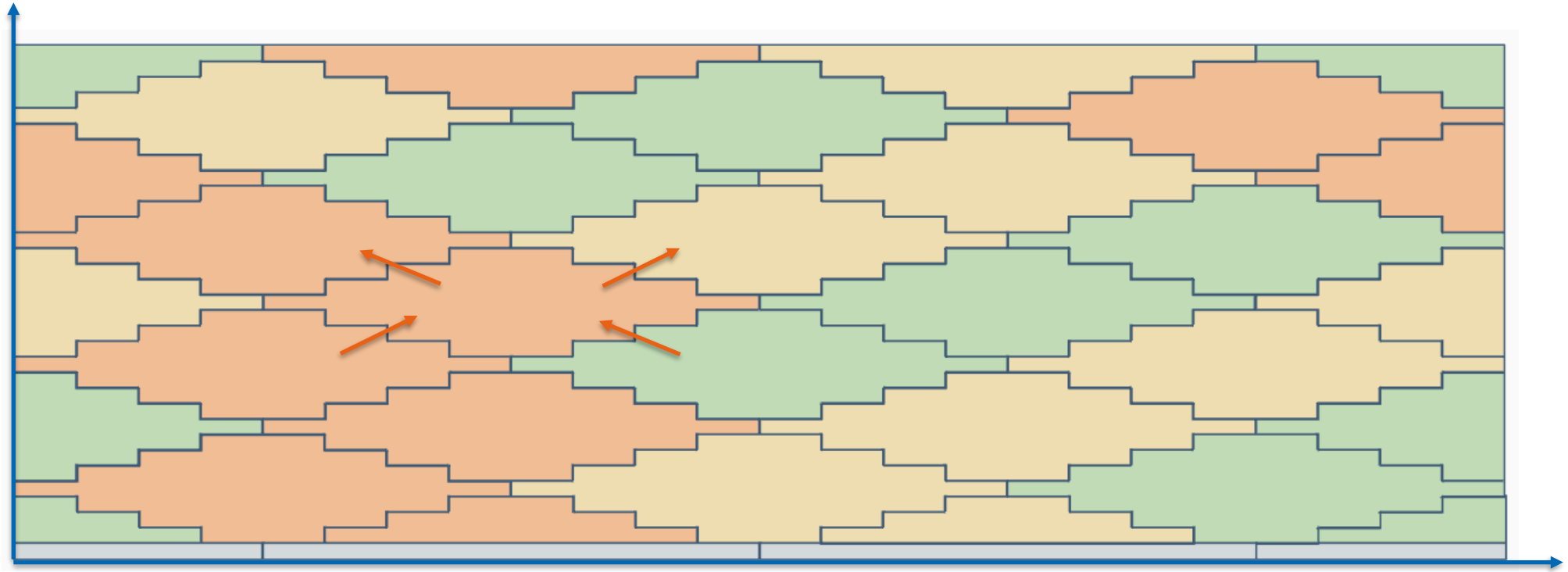
- Stand for **M**ulticore **W**avefront **D**iamond tiling **T**emporal **B**locking technique.
- Promote different strategies for each dimension:
  - X : Vectorization (SIMD).
  - Y : Diamond tiling.
  - Z : Wavefront parallelism.

- Expose key architectural advantages:
  - Reduce memory bandwidth pressure.
  - Enable natural overlapping of computation and communication.

- References:
  - *Tiling and Asynchronous Communication Optimizations for Stencil Computations,* Tareq Malas, PhD thesis, KAUST, 2015.
  - *Multicore-Optimized Wavefront Diamond Blocking for Optimizing Stencil Updates*, T. Malas, G. Hager, H. Ltaief, H. Stengel, G. Wellein & D. Keyes, SIAM J. Sci Comput. 37:C439-C464, 2015.

MWD-TB background

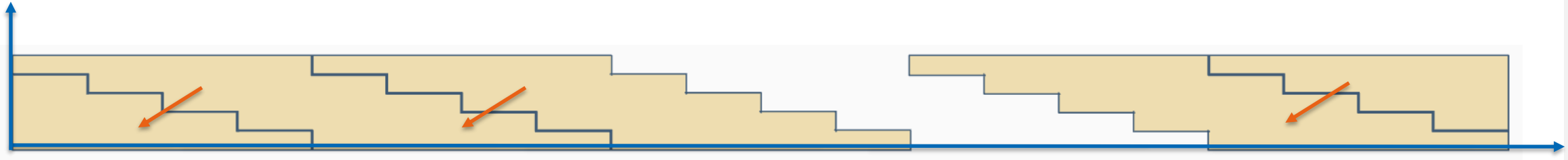
- Diamond tiling
  - Outer-level OpenMP with dynamic scheduling on different groups of threads.



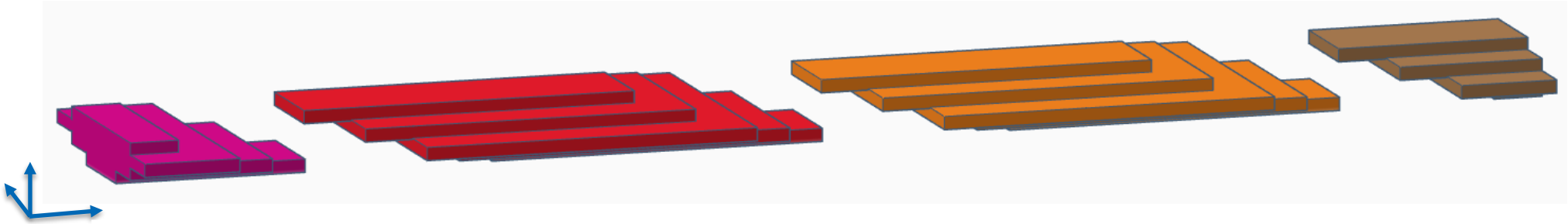
Diamond tiling and the dependency among diamonds
(color represents three diff. groups of threads)

MWD-TB background

- Wavefront parallelism
  - Inner-level OpenMP.
  - cache reuse among threads and between contiguous wavefront steps.



Wavefront and the dependency among different wavefront steps

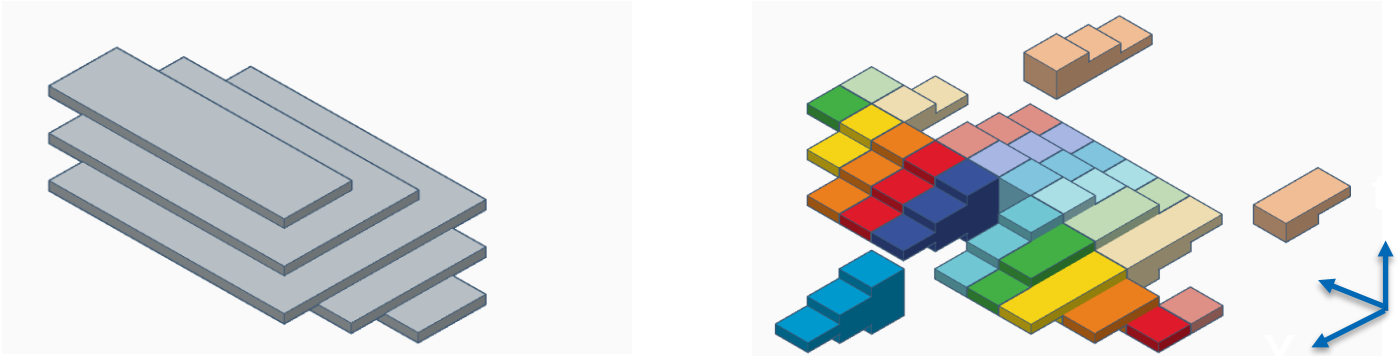


3D views of different wavefront steps

Qu L, Abdelkhalak R, Ltaief H, Said I, Keyes D. Exploiting temporal data reuse and asynchrony in the reverse time migration. The International Journal of High Performance Computing Applications. 2023;37(2):132-150. doi:10.1177/10943420221128529

MWD-TB stencil computation

- Multi-core computation of one wavefront step
  - cache reuse inside each wavefront step.
  - inner level OpenMP barrier between each time step.

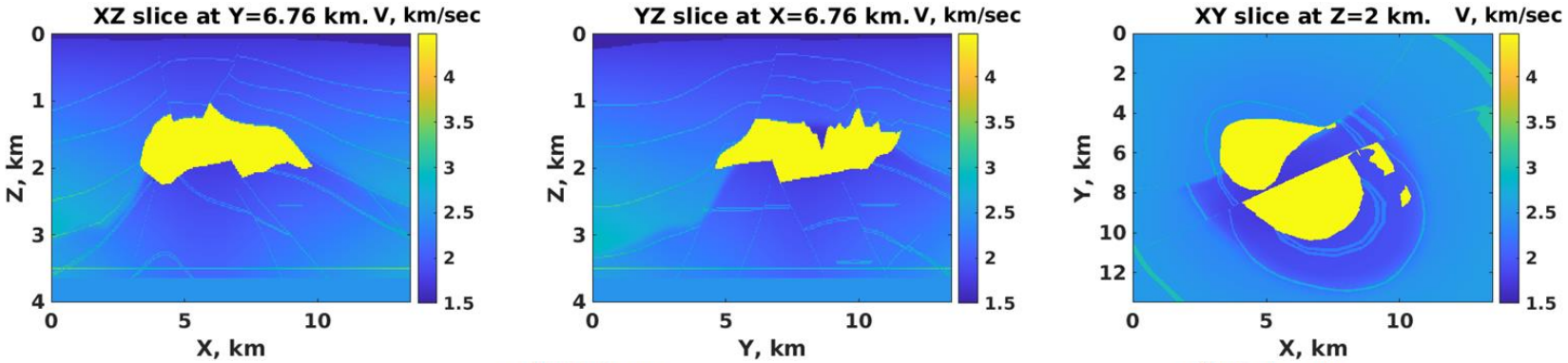


3D view of one wavefront step
inside a diamond

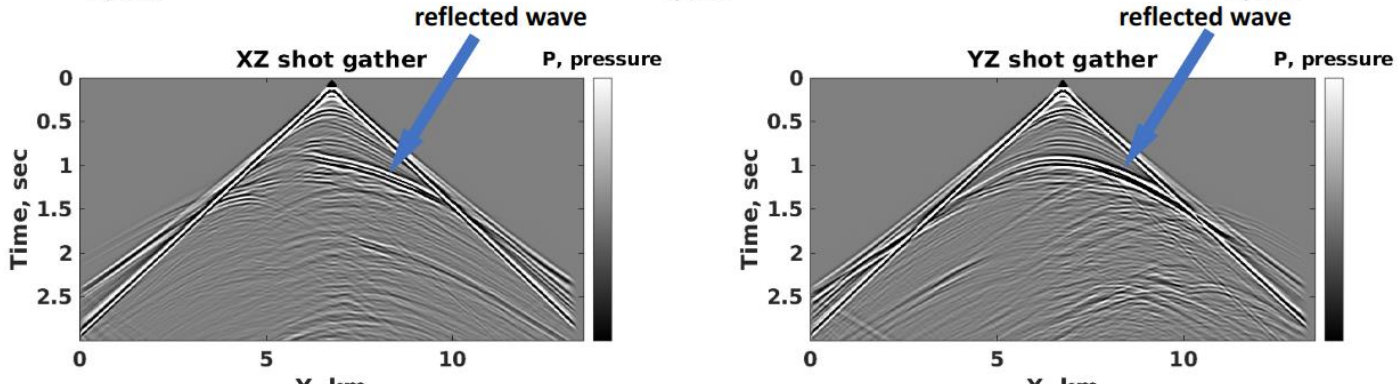
Qu L, Abdelkhalak R, Ltaief H, Said I, Keyes D. Exploiting temporal data reuse and asynchrony in the reverse time migration. The International Journal of High Performance Computing Applications. 2023;37(2):132-150. doi:10.1177/10943420221128529

Seismic forward modelling problem

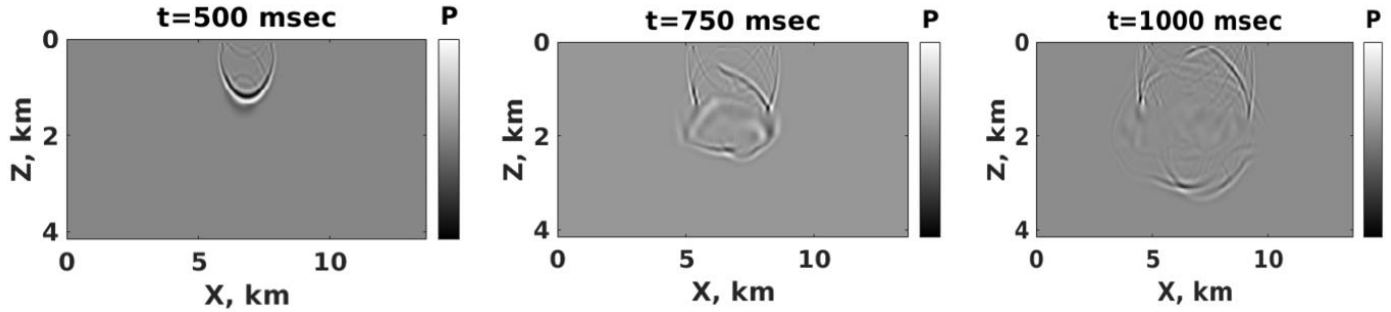
Seismic waves velocity



Seismograms



Snapshots



Outline

- Stencil computations
  - Spatial blocking
  - Temporal blocking
- Memory Hierarchy and Flow
  - Cache coherence
- Performance results

Cache Coherence

- There are two levels of coherence in Intel® Xeon® architecture.
  - Coherence between the L1/L2 of each core.
  - Coherence between caches of each socket.

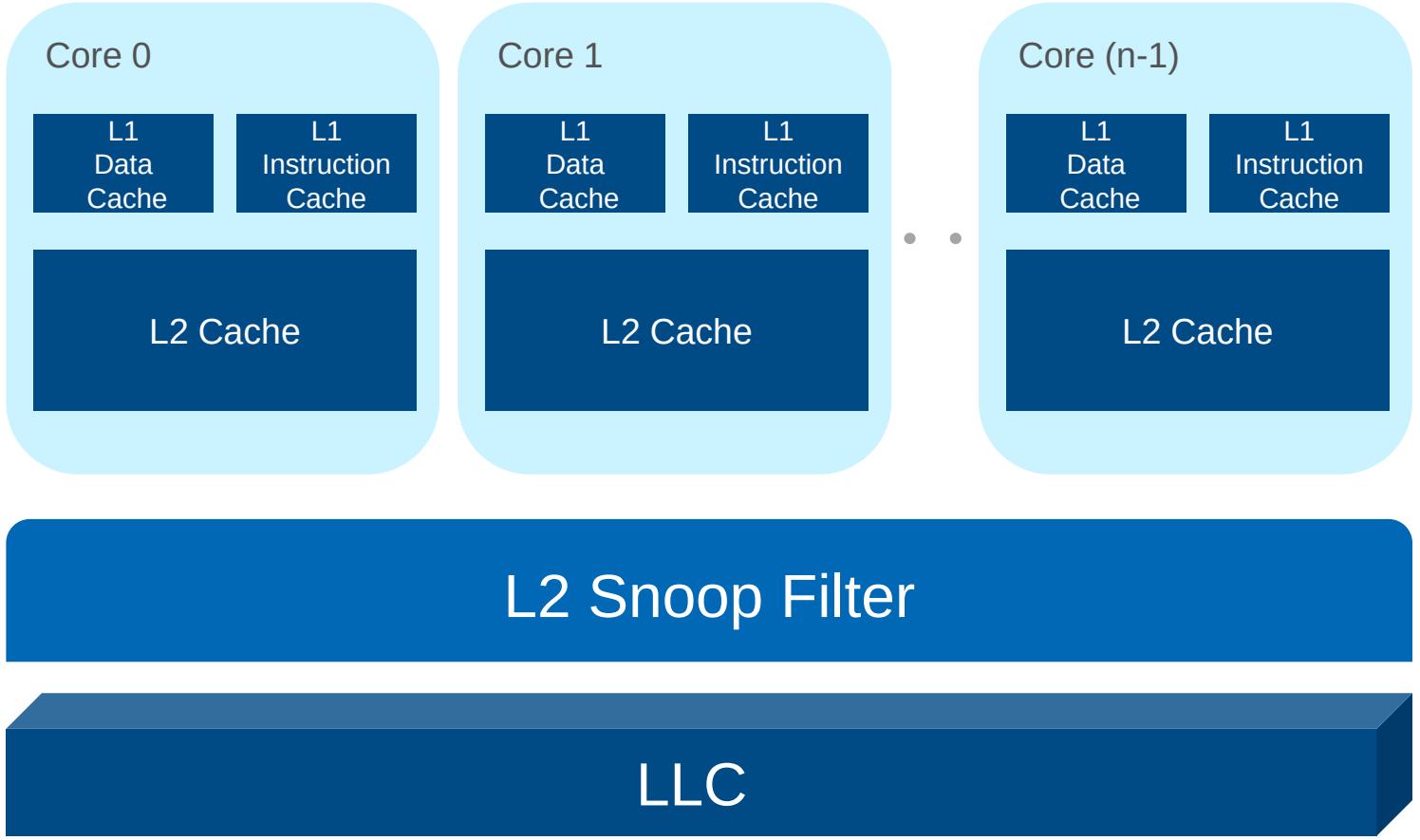
- Coherence is maintained by:
  - Snoop filter – keeps track of which cores have the line.
  - Directory – keeps track of which sockets have the line.
  - CHA (Caching and Home Agent) – the logic to handle all the above.

Local Coherence (within a socket)

- L1/L2 is private to the core and cannot be accessed by other cores unless it is snooped.
- Non-Inclusive Last Level Cache.
  - LLC is shared between all local cores.
  - A line in the L1/L2 may not necessarily be in the LLC (non-inclusive does not mean exclusive).
  - Generally, caches data that are no longer resident in the core caches.

Local Transaction Flows (within a socket)

When a core attempts to access a specific address line, what are the possible scenarios?

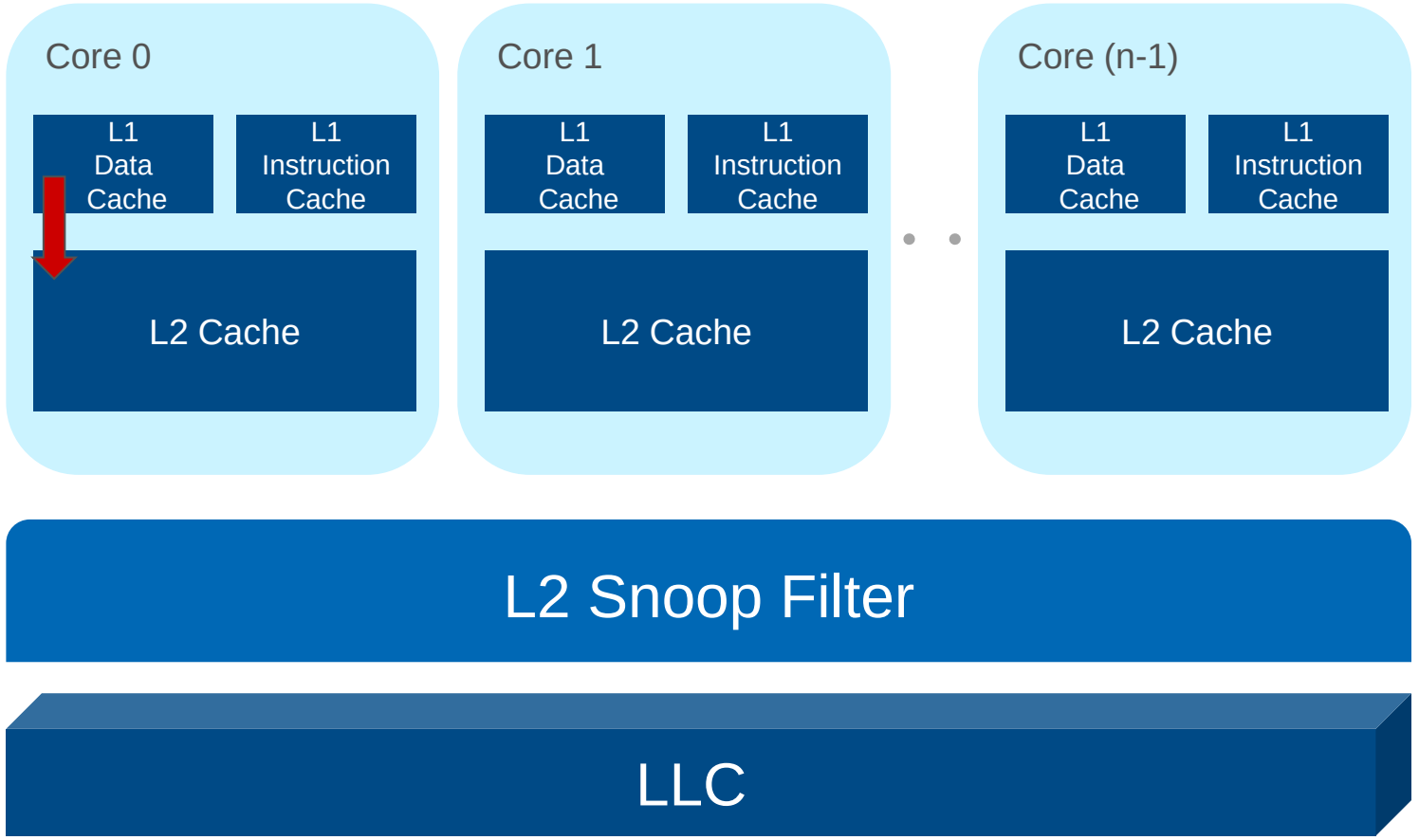


Local Transaction Flows (within a socket)

When a core attempts to access a specific address line, what are the possible scenarios?

1. First, the core checks if the address is already in its own **L1/L2**

   If so, core just grabs the line out of the L1/L2 and uses it. This is the best-case scenario, latency-wise -> very fast.



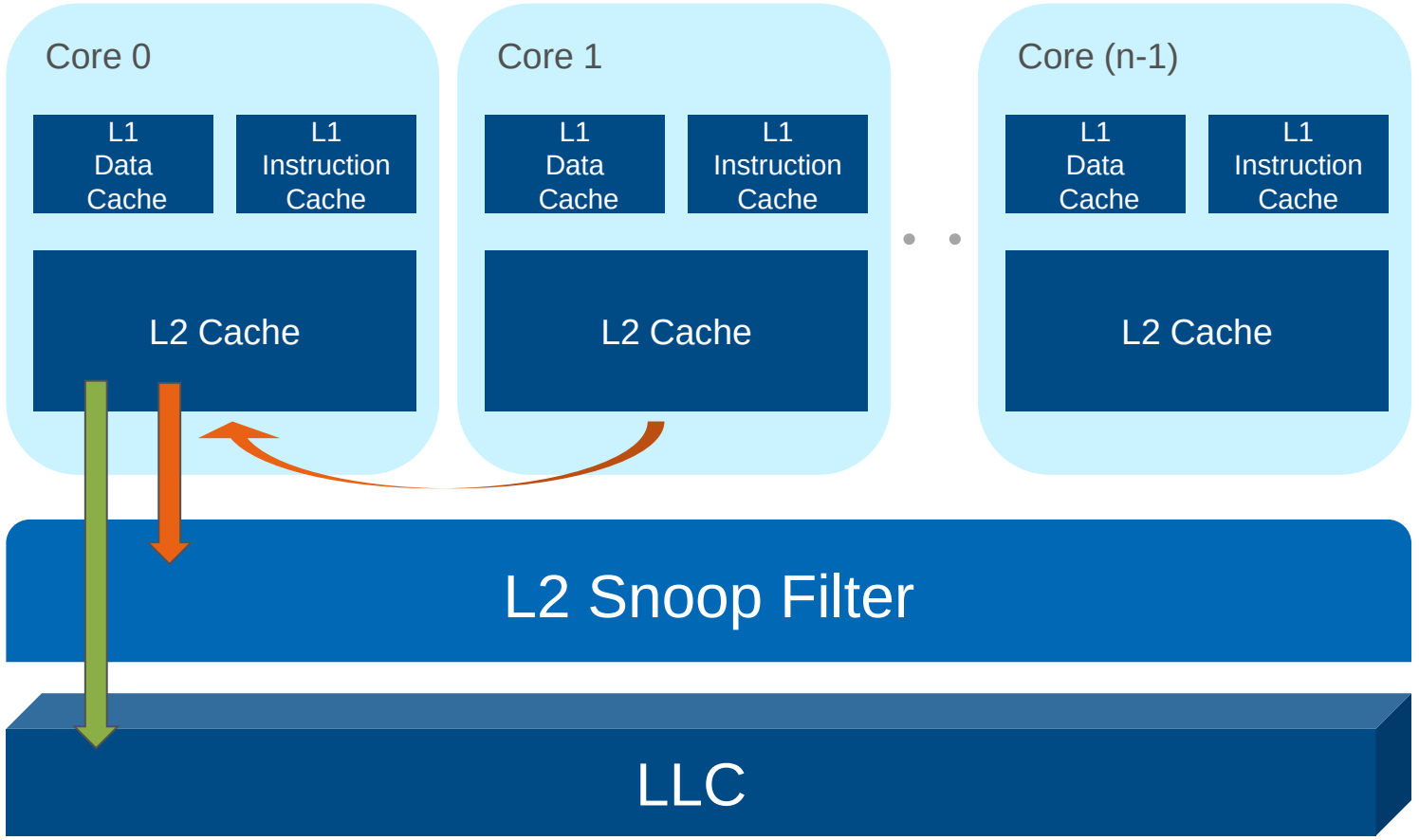
Local Transaction Flows (within a socket)

When a core attempts to access a specific address line, what are the possible scenarios?

2. Second, if not in the core's own L1/L2? need to check:
   1) If it's in another core's L1/L2 *or*
   2) If it's in the LLC.

The core sends out a message to the CHA. The CHA will look up the address in the **snoop filter** and in the **LLC**.

- A snoop filter hit means the data is in a different core's L1/L2. It sends a snoop to that core, and it returns the data to the requesting core.
- If it miss in the snoop filter but hit in the L3, then the CHA returns the data to the core.



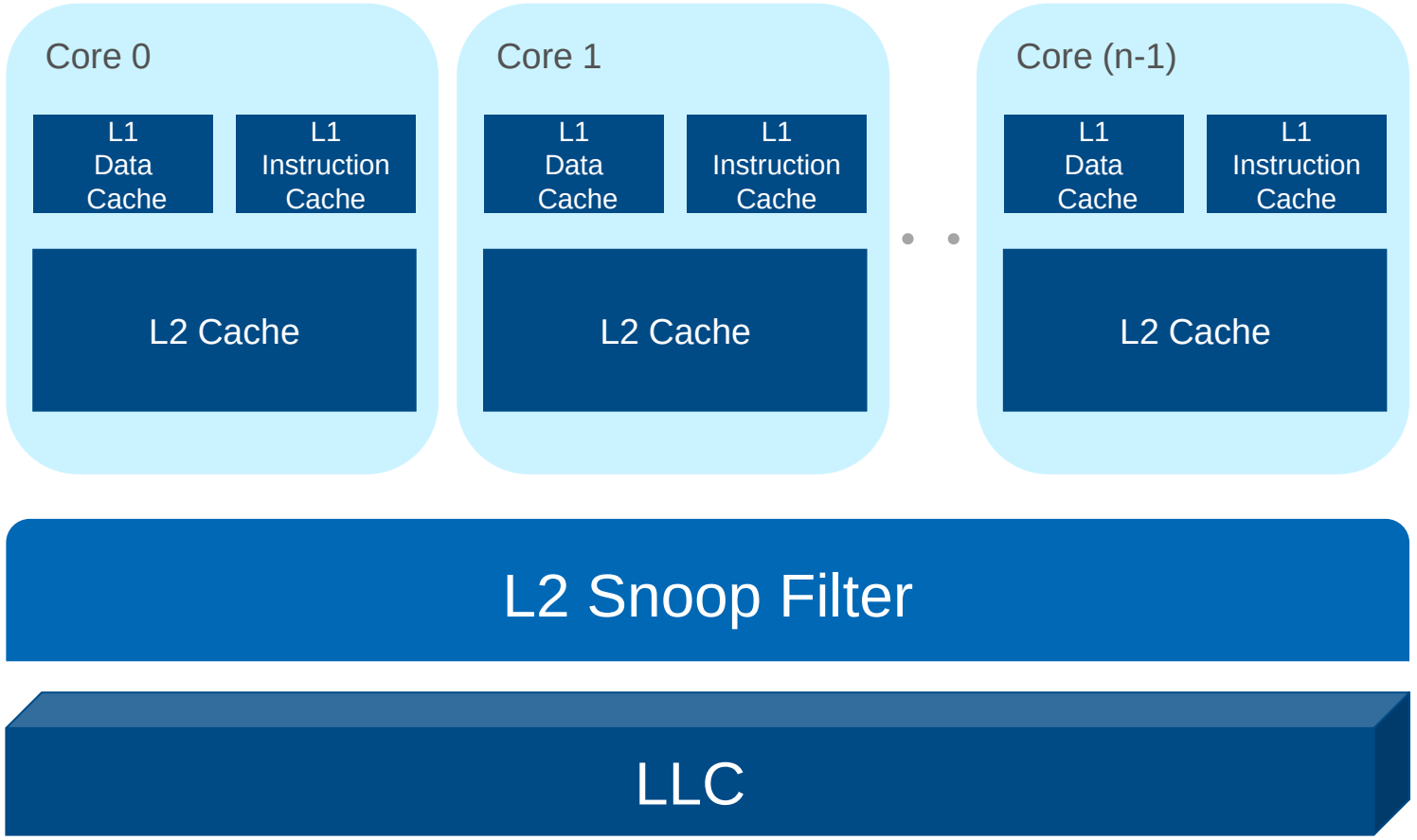
The Snoop Filter (SF) is a structure in the CHA that tracks which cores have the line.

Local Transaction Flows (within a socket)

When a core attempts to access a specific address line, what are the possible scenarios?

3. Not in any caches

   Need to go to **DRAM** to get the data.

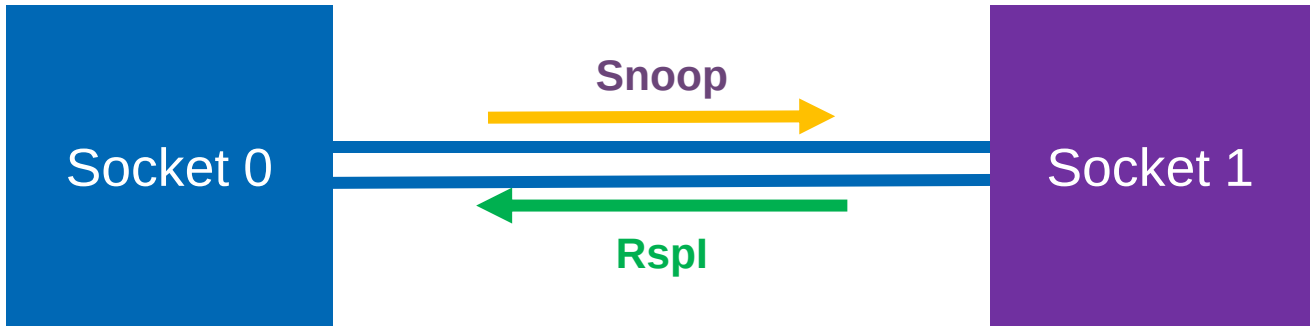


Global Coherence (between sockets)

- Manages coherency, consistency, ordering of accesses to cache lines between sockets.
- Directory records where and in what state each memory line is cached in.
- Directory is maintained in DRAM as part of the actual line.
  - A few bits in memory are stolen to maintain directory.
  - One read to DRAM can return both directory info and the actual data.
  - Both data and directory can be updated with a single DRAM write.
- Three directory states stored in two bits.

Directory state	Description
Invalid (I)	Line not present in any other socket, but local socket can have the line cached.
Any (A)	Line may be present in any socket (snoop needed to ensure coherence).
Shared (S)	Line may be present in one or more sockets, but the line can only be in shared unmodified state – no snoop needed for reads. Snoop needed to modify the line.

Global Transaction Flows (DDR)



RspI (response invalid) is a snoop response meaning “I don’t have the data you were looking for”

1. Core on socket0 sends a read request to CHA.
2. Miss in SF/LLC
3. Data is retrieved from CHA sends a request to DRAM
4. DRAM and sent back to CHA
5. CHA sees that directory=A state!
6. CHA sends snoop over UPI to socket1

7. Socket1 receives the snoop over UPI. It goes to the CHA.
8. Miss in SF/LLC
9. CHA sends RspI over UPI to socket0

What if socket 1 would have had a SF/LLC hit? It would have forwarded the data instead of sending RspI.

10. CHA receives the snoop response. Because it was RspI, it knows the data it already got from DRAM is clean

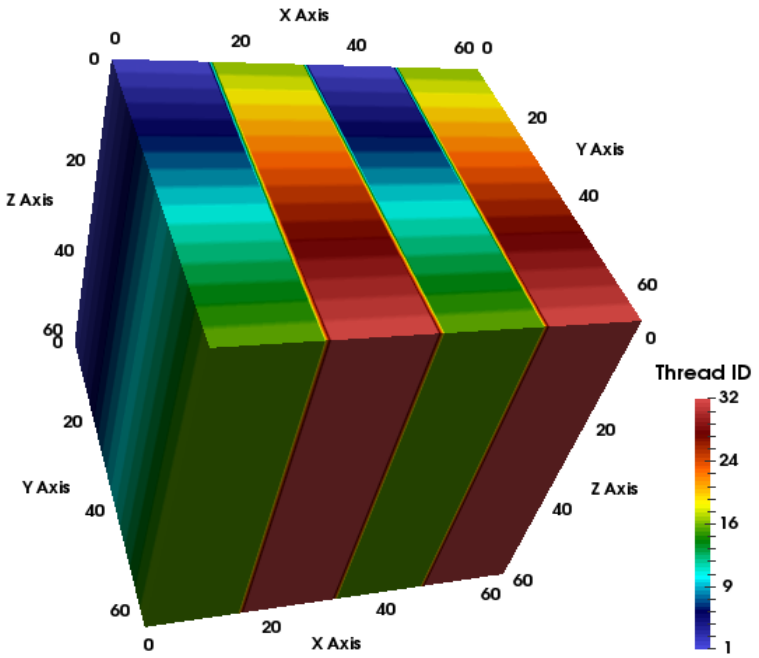
Global Transaction Flows (HBM)

- HBM doesn't support directory -> use **Remote Snoop Filter (RSF)**.
- **Challenge:** when Socket B accesses Socket A memory and LLC/SF of Soc B > RSF of Soc A.
- When socket B reads line L, socket A needs to create an entry in its RSF to track the remote access.
- However, to create a new entry, the RSF of socket A must sometimes evict an entry that is already present in the RSF, because RSF has a limited size and hence can track only a limited number of lines at a given time.
- If enough unique cache lines are transferred to overflow the capacity of the RSF, socket A must send a message to socket B to invalidate the line associated with the victim entry in the RSF.
- Therefore, **LLC effective size is limited by the RSF size**.

Outline

- Stencil computations
  - Spatial blocking
  - Temporal blocking
- Memory Hierarchy and Flow
  - Cache coherence
- Performance results

Spatial Blocking (SB)

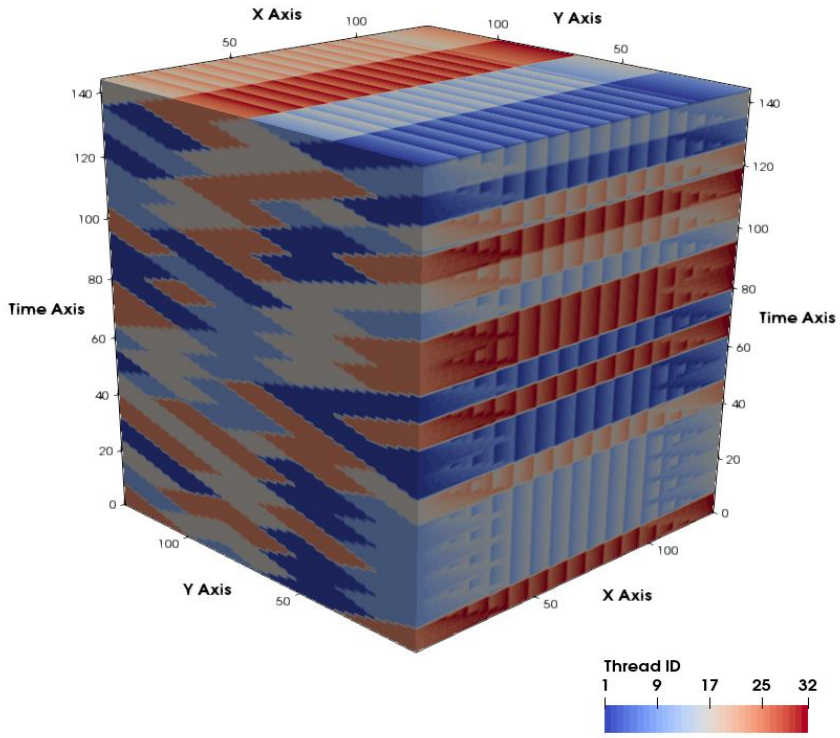


Memory-bound.

```
1:  procedure SB(P, nt, lx, ly, lz, bsx, bsy, th)
2:      ▷ nt: number of time steps
3:      ▷ l{x, y, z}: length of X, Y and Z dimensions
4:      ▷ bs{x, y}: block size for X and Y dimensions
5:      ▷ th: number of threads
6:      for n ← 1 to nt do                                  ▷ time loop
        ▷ Iterations of the consecutive two parallel "for" loops
        are processed by th threads.
7:          parallel for x←1 to lx step bsx do              ▷ blocking in X
8:              parallel for y←1 to ly step bsy do          ▷ blocking in Y
9:                  for i ← (x − 1) bsx to x bsx do
10:                     for j ← (y − 1) bsy to y bsy do
11:                         for k ← 0 to lz − 1 do
12:                             update P[i][j][k]
```

Akbudak K, Ltaief H, Etienne V, Abdelkhalak R, Tonellot T, Keyes D. Asynchronous computations for solving the acoustic wave propagation equation. The International Journal of High Performance Computing Applications. 2020;34(4):377-393. doi:10.1177/1094342020923027

Temporal Blocking (TB)

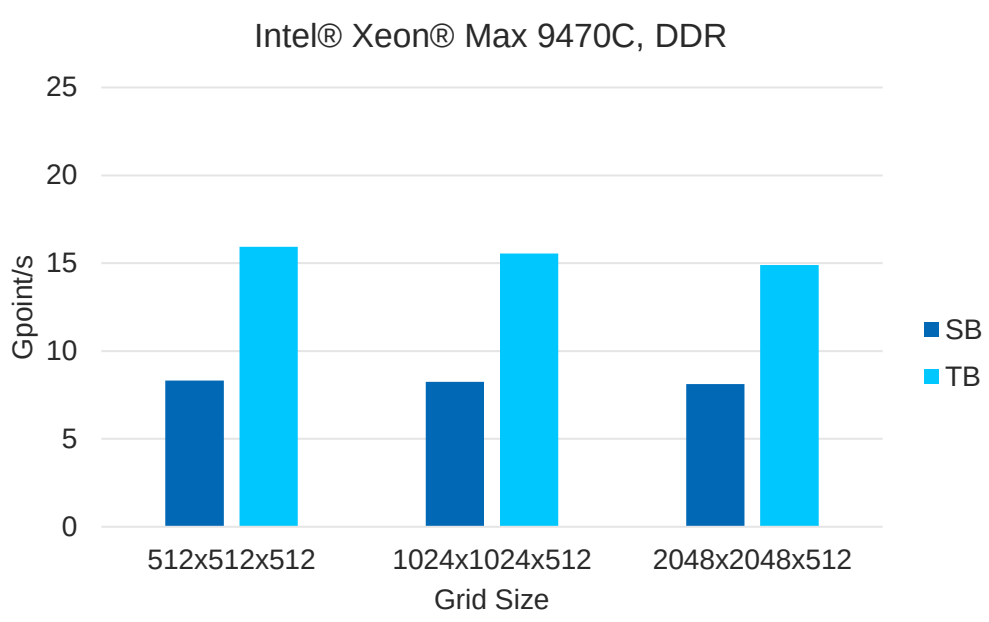


LLC-bound.

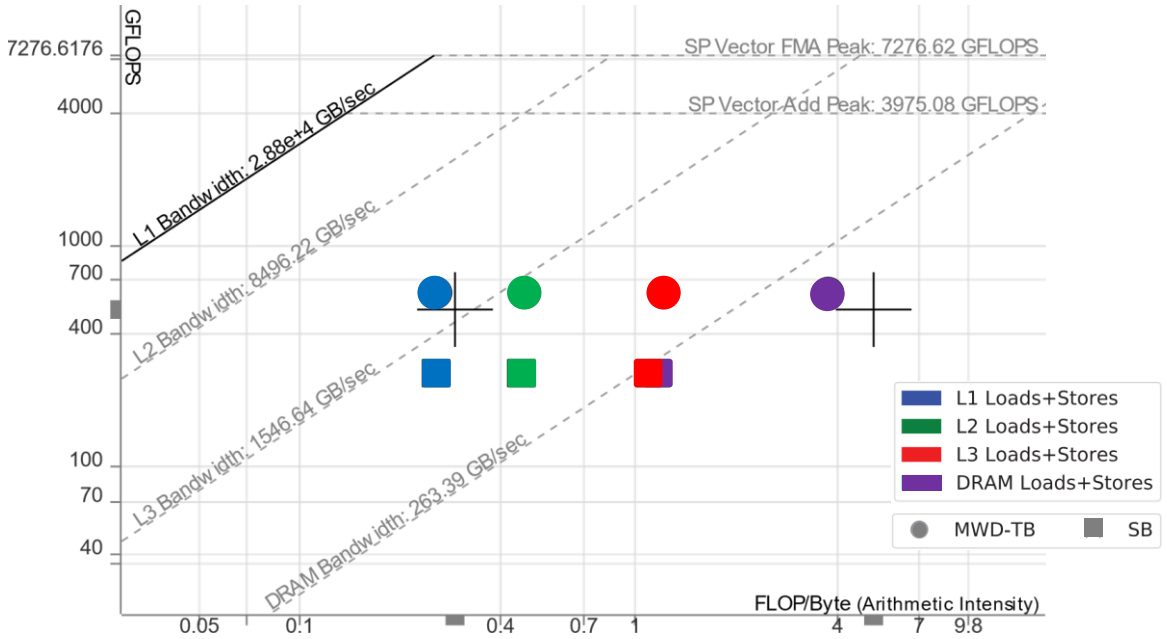
```
1: procedure MWD-TB-MAIN(P, nt, lx, ly, lz, th, th_x, nwf, th_nwf)
2:     ▷ nt: number of time steps
3:     ▷ l{x, y, z}: length of X, Y and Z dimensions
4:     ▷ th: total number of threads
5:     ▷ th_x: number of threads in x dimension
6:     ▷ nwf: size of wavefront block
7:     ▷ th_nwf: size of wavefront per thread
8:   in parallel with th/th_x threads do
9:     for each block blk in Y dimension do
10:      if dependent blocks are finished then
11:        Update P to setup the wavefront in X using SB
12:        MWD-TB()  ▷ process blk and proceed in time
13:        Update P to complete the wavefront in X using SB

   ▷ Processes one diamond-shaped block in time and Y dimension domain.
14: procedure MWD-TB(P, tb, te, xb, xe, yb, ye, zb, ze, th_x, nwf, th_nwf)
15:    ▷ tb, te: beginning&end of time loop
16:  ▷ {x, y, z}{b, e}: beginning&end indices of x, y and z
17:    ▷ th_x: number of threads in x dimension
18:    ▷ nwf: size of wavefront block
19:    ▷ th_nwf: size of wavefront per thread
20:  in parallel with th_x threads do
21:    for xi ← xb to xe by nwf do ▷blocking in wavefront
22:      for n ← tb to te do  ▷ time loop
23:        for i ← xi to xi + nwf do
24:          if (i/th_nwf)%th_x == tid_x then
25:            for j ← yb to ye do
26:              for k ← zb to ze do
27:                update P[i][j][k]
```

Akbudak K, Ltaief H, Etienne V, Abdelkhalak R, Tonellot T, Keyes D. Asynchronous computations for solving the acoustic wave propagation equation. The International Journal of High Performance Computing Applications. 2020;34(4):377-393. doi:10.1177/1094342020923027

Performance, 1 socket (DDR, Intel® Xeon® Max 9470C)



TB achieves up to 2X speed-up compared to SB.

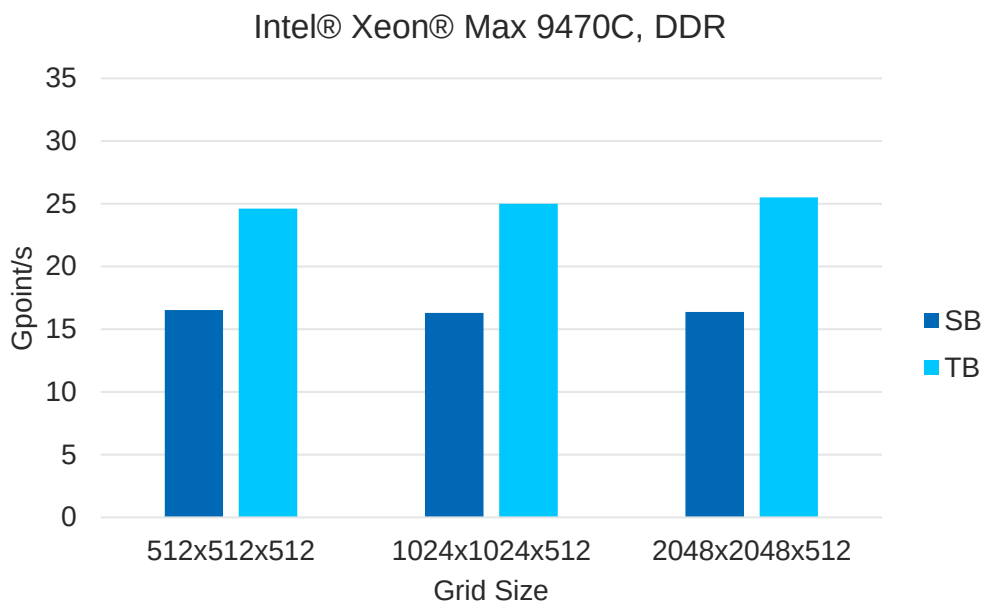


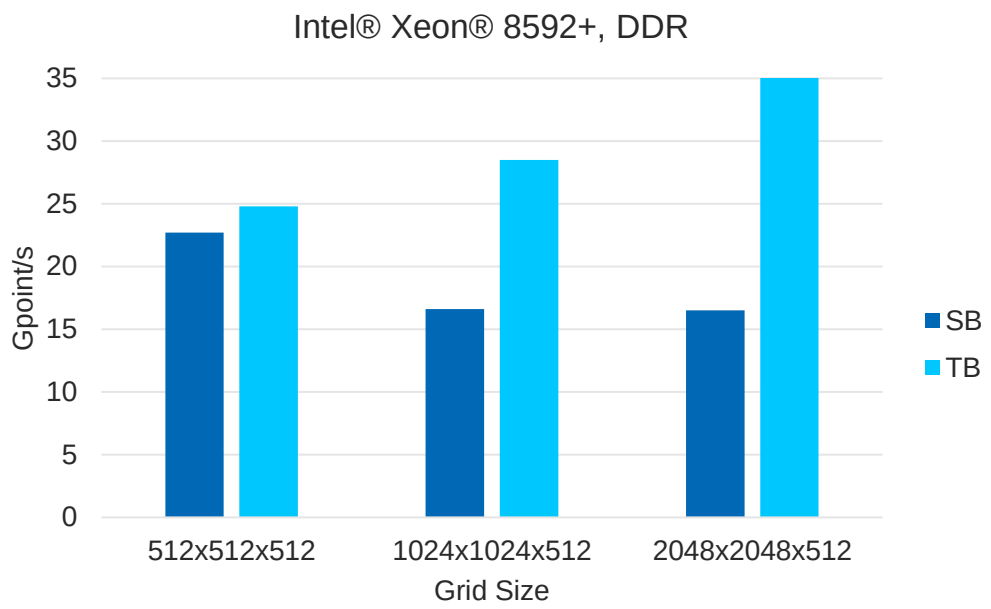
In TB, the wider gap between L3 and DRAM bandwidths compared to SB confirms that L3 exploits data locality more, resulting in less DRAM traffic.

See backup for workloads and configurations. Results may vary.

Performance (DDR, Intel® Xeon® Max 9470C vs. Intel® Xeon® 8592+)

2-sockets	Intel® Xeon® Max 9470C	Intel® Xeon® 8592+
#cores	104	128
LLC (MB)	210	640
DDR5 (GB)	1024	1024

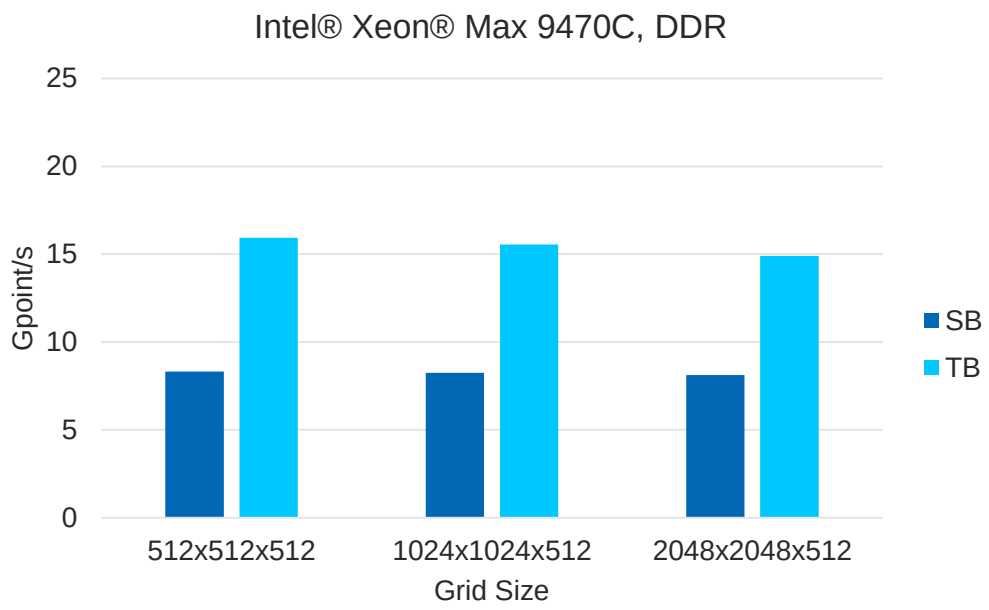




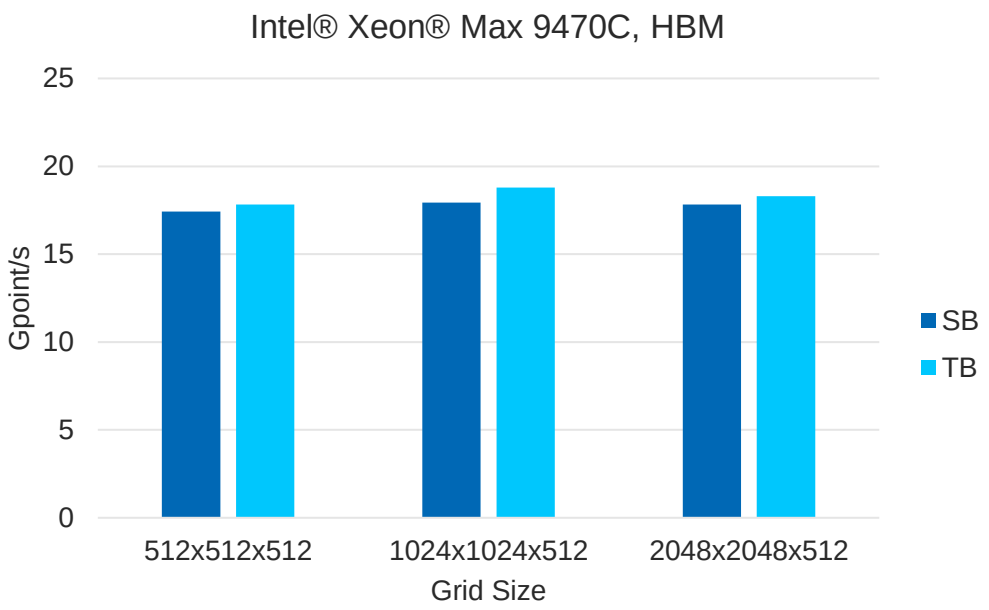
TB can better leverage the large LLC capacity of Intel® Xeon® 8592+ compared to SB.

See backup for workloads and configurations. Results may vary.

Performance, 1 socket (DDR vs. HBM on Intel® Xeon® Max 9470C)



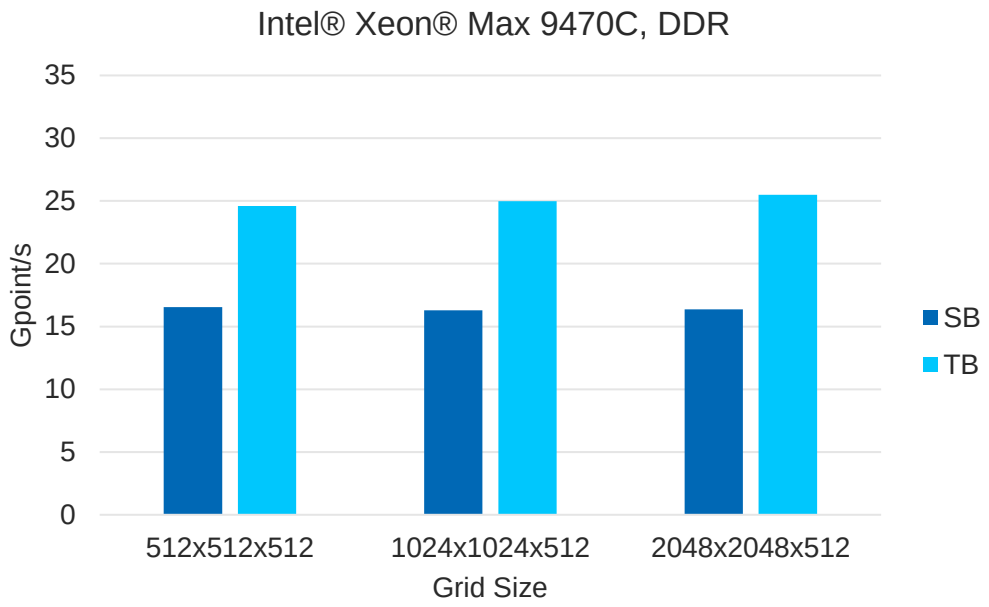
SB achieves up to over a 2X speed-up on HBM compared to DDR.



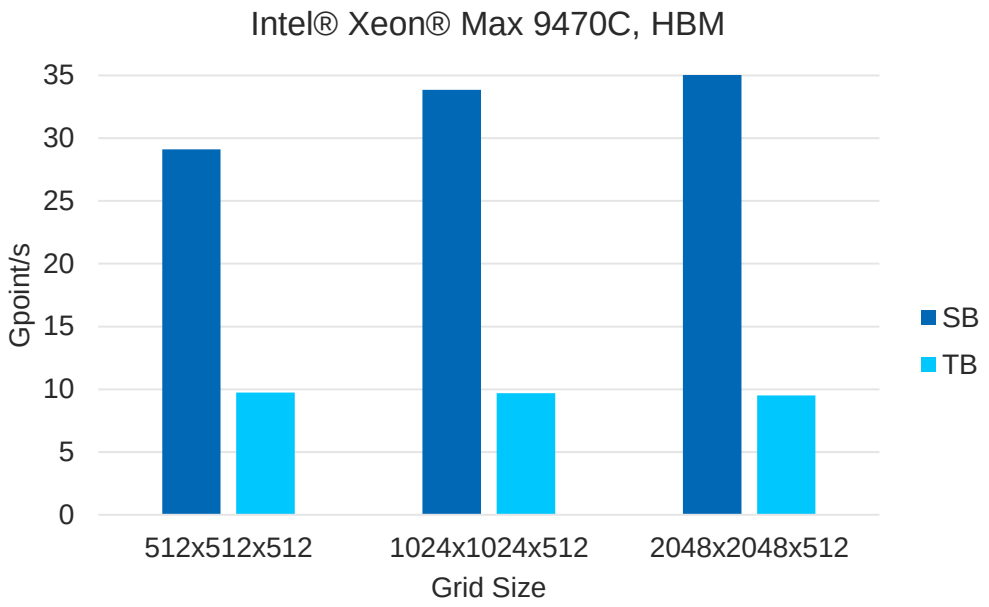
As TB is LLC-bound, HBM doesn't offer significant benefits.

See backup for workloads and configurations. Results may vary.

Performance, 2 sockets (DDR vs. HBM on Intel® Xeon® Max 9470C)



SB achieves up to over a 2X speed-up on HBM compared to DDR.



TB's performance drops when using HBM on two sockets compared to one socket, as it becomes bound by the RSF.

See backup for workloads and configurations. Results may vary.

Solution: Improve Locality

- Use multiple of two MPI ranks.
- If using OpenMP across NUMA domains, be NUMA-aware.
    - E.g., first-touch within OMP region.

Notices & Disclaimers

- Performance varies by use, configuration and other factors. Learn more on the Performance Index site.
- Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. No product or component can be absolutely secure.
- Your costs and results may vary.
- Intel technologies may require enabled hardware, software or service activation.
-

System Configurations and Performance Data Sources

- **SB and TB:** Test by Intel as of 05/09/2024, two sockets Intel® Xeon® CPU Max 9470C formerly Sapphire Rapids HBM , SUSE Linux Enterprise Server 15 SP4, oneAPI pre-production software.
- **SB and TB:** Test by Intel as of 05/09/2024, two sockets 5th Generation Intel® Xeon® Scalable Processor Platinum 8592+, SUSE Linux Enterprise Server 15 SP5, oneAPI 2024.1.0 software.

Thank you.