Global
Address Space
Programming Interface
GASPI

www.gaspi.de

GASPI-SHAN

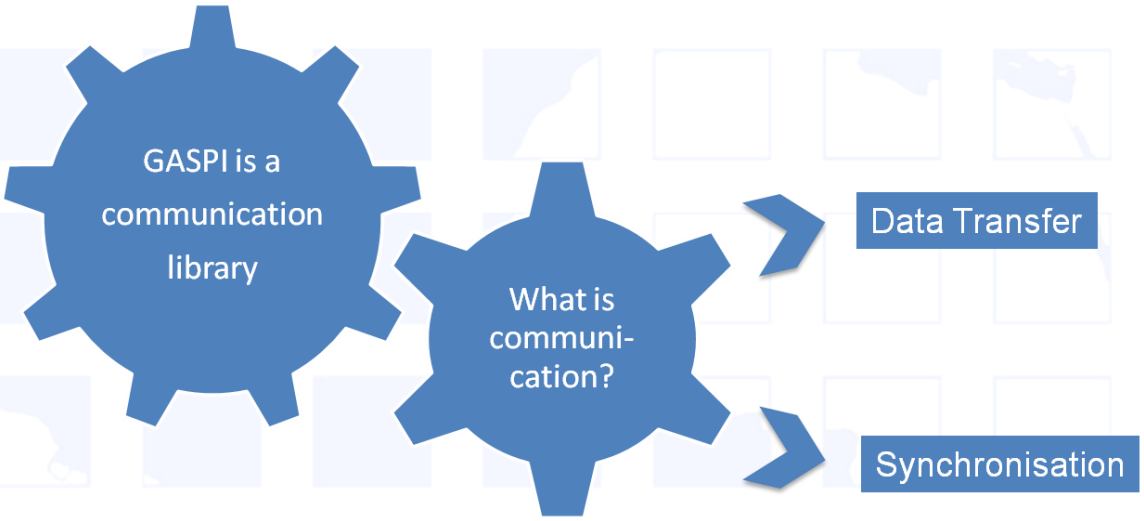
Christian Simmendinger

T-Systems SfR

Sponsored by the European Commission through

Global
Address Space
Programming Interface
GASPI

GASPI at a Glance



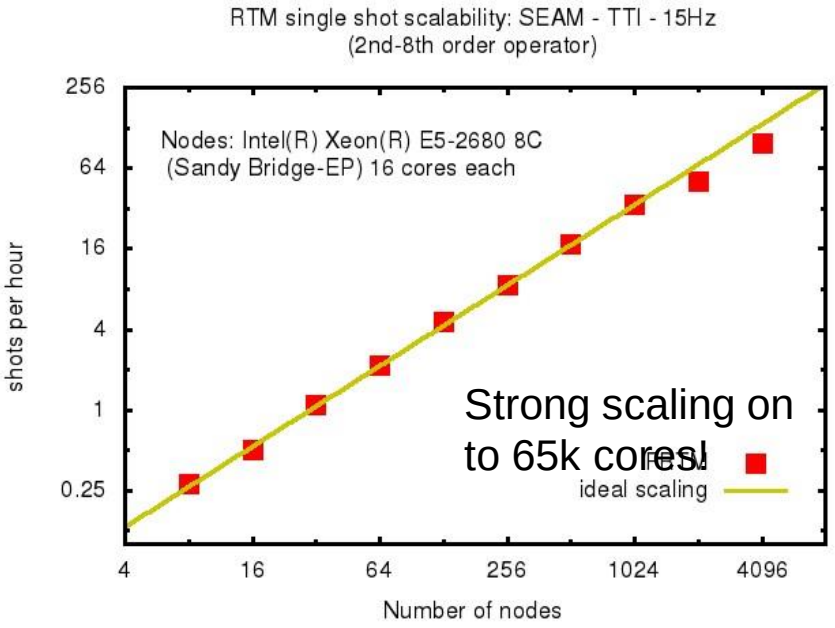
Nuts and Bolts for Communication Engines

Global
Address Space
Programming Interface
GASPI

GASPI at a Glance

Features:

- Global Partitioned Address Space
- Thin abstraction layer, designed for
  - Asynchronous, one-sided (bundled) communication
  - Multithreaded execution.
  - Failure tolerance
- Standardized API (GASPI), Open Source
- Commercial support (GPI2, Fraunhofer)



Infiniband, Cray, Roce Ethernet, TCP, GPUs, Intel64, Intel Xeon Phi, ARMv8

Global
Address Space
Programming Interface
GASPI

GASPI
Standardization Forum

Founding
Members

Global
Address Space
Programming Interface
GASPI

GASPI in European Exascale Projects

EXascale Algorithms and Advanced Computational Techniques

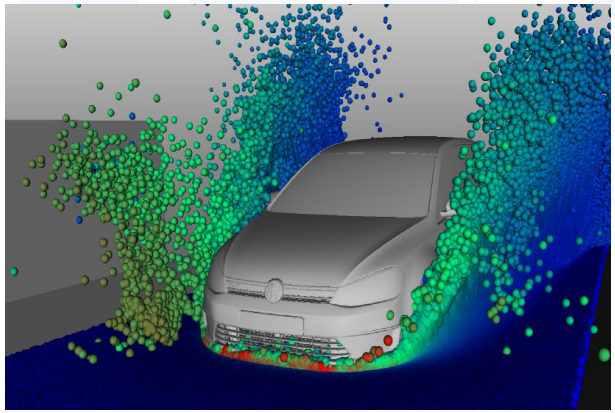
Programming-model design and implementation for the Exascale

Exascale ProGRAmming Models

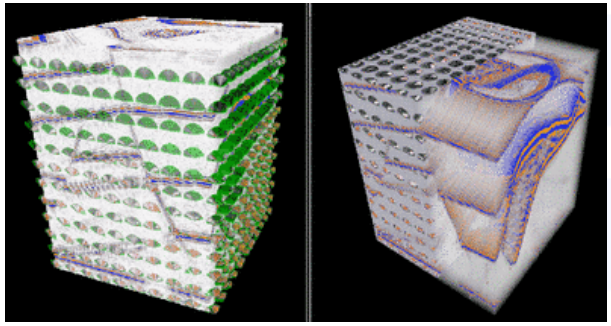
Global
Address Space
Programming Interface
GASPI

Some GASPI Applications

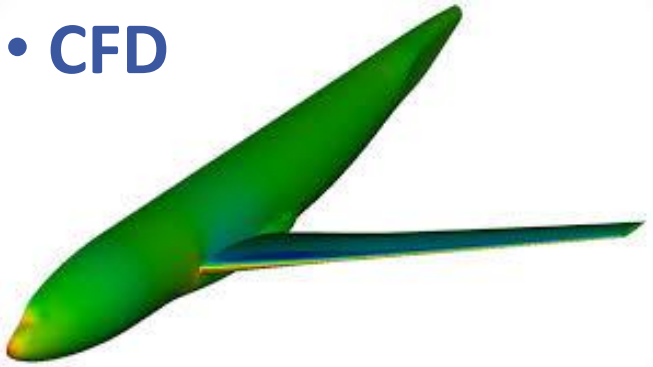
- Visualization



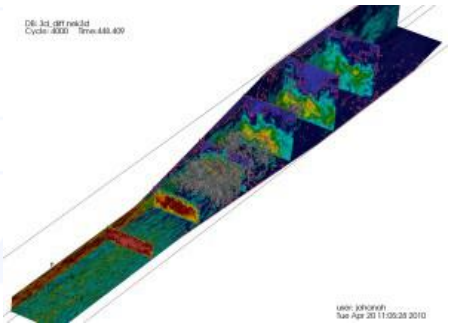
- Seismic Imaging & Algorithms



- CFD



- Machine Learning
- Big Data
- Iterative Solvers



Global
Address Space
Programming Interface
GASPI

Concepts: Communication

GASPI is a communication library

What is communi-cation?

Data Transfer

- Memory Segments

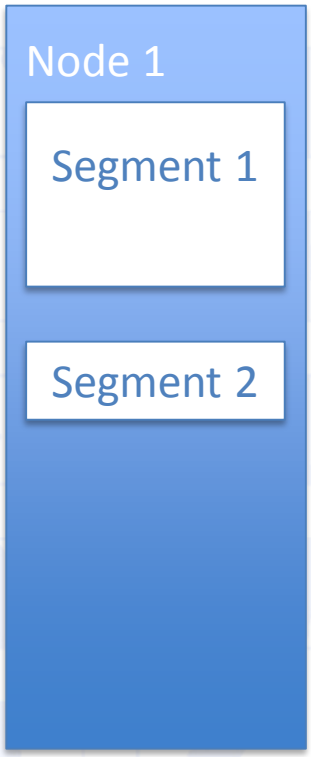
Synchronisation

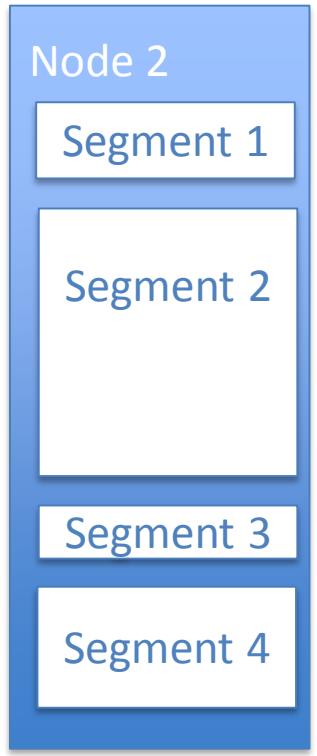
- Notified Communication
- Remote Completion
- Local Completion

Global
Address Space
Programming Interface
GASPI

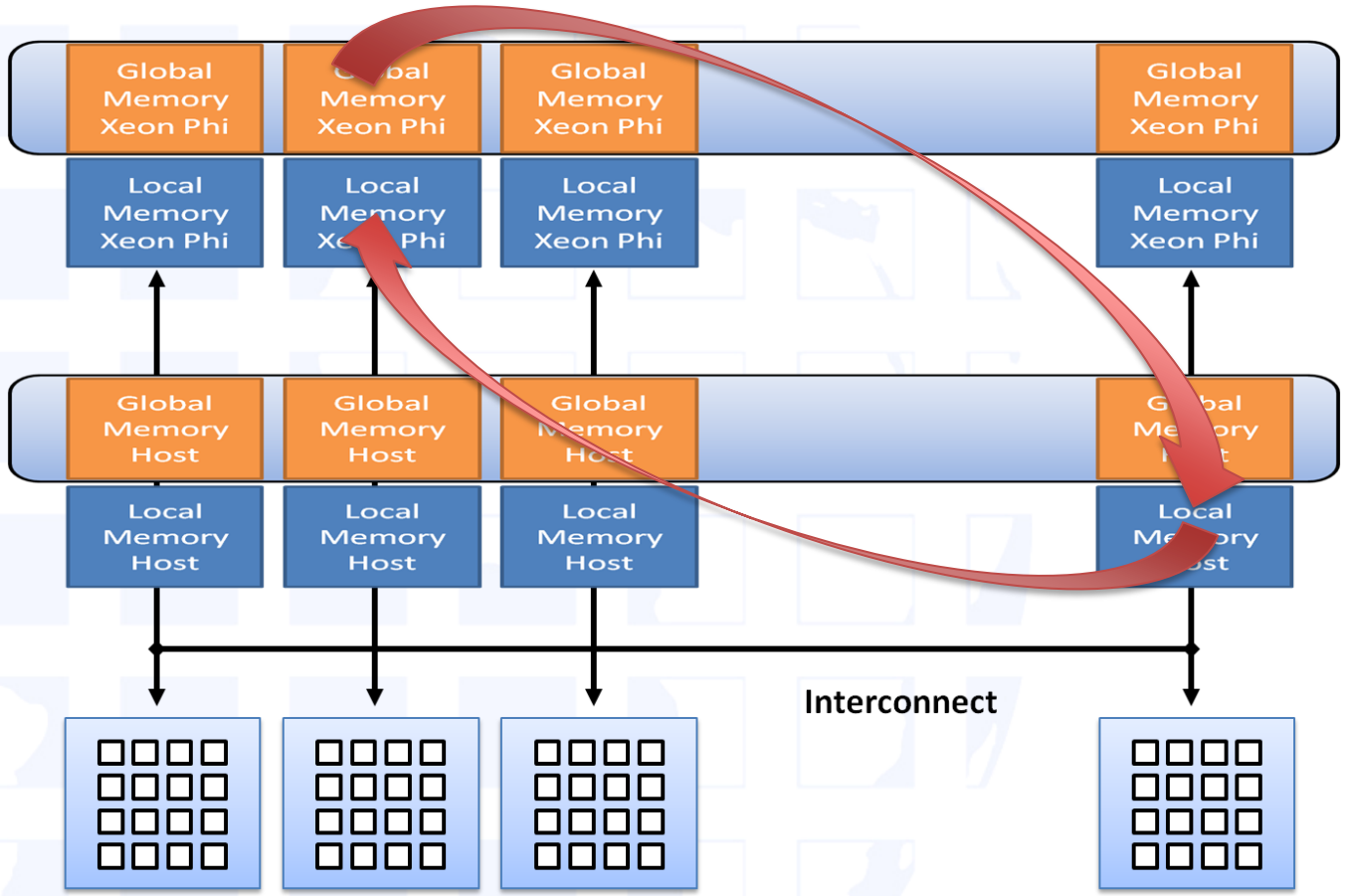
Memory Segments

- Designated communication areas in GASPI are called memory **segments.**
- Segments are freely configurable.
- No limitations to memory model. (e.g data parallel)
- Data can be accessed without participation of the remote site.





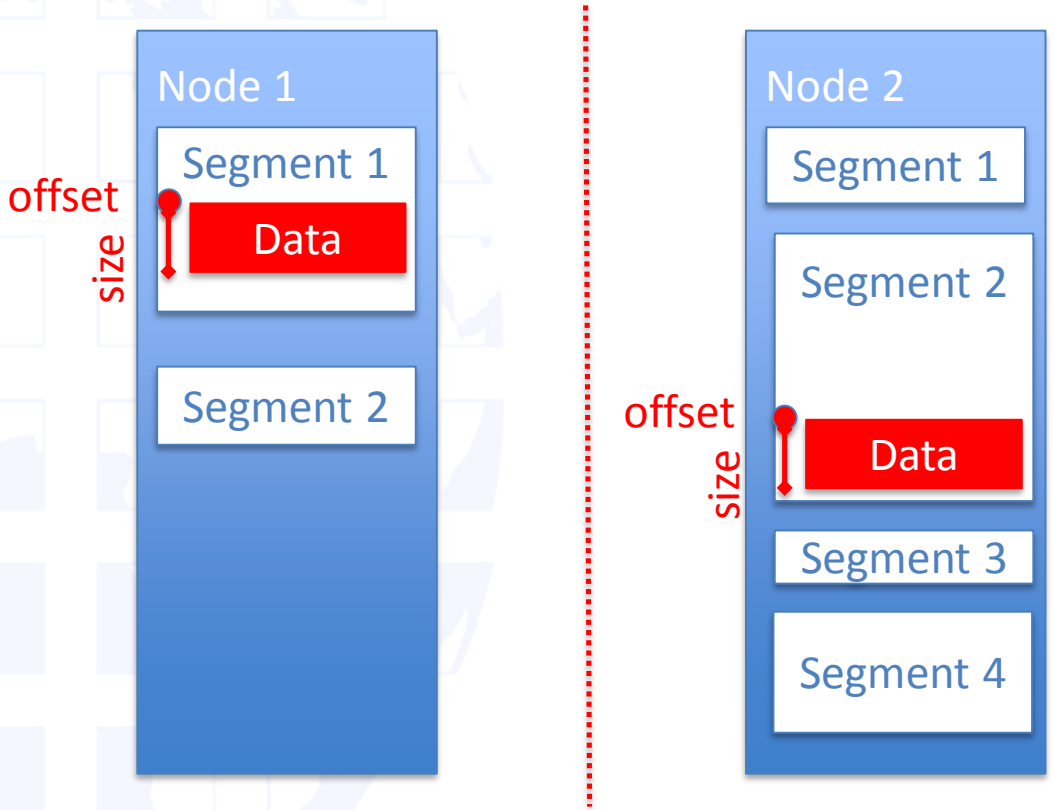
Memory Segments



Memory Segments

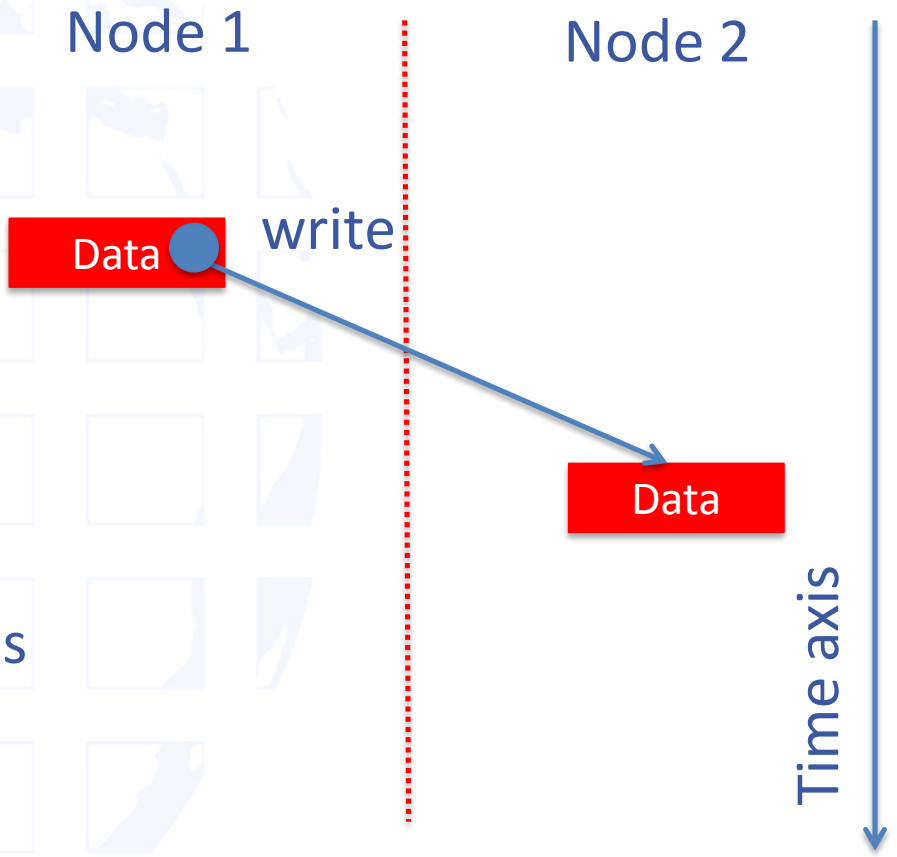
Application has to manage the data transfer.

- Specify which part of the segment will be transferred
- **Offset and Size**



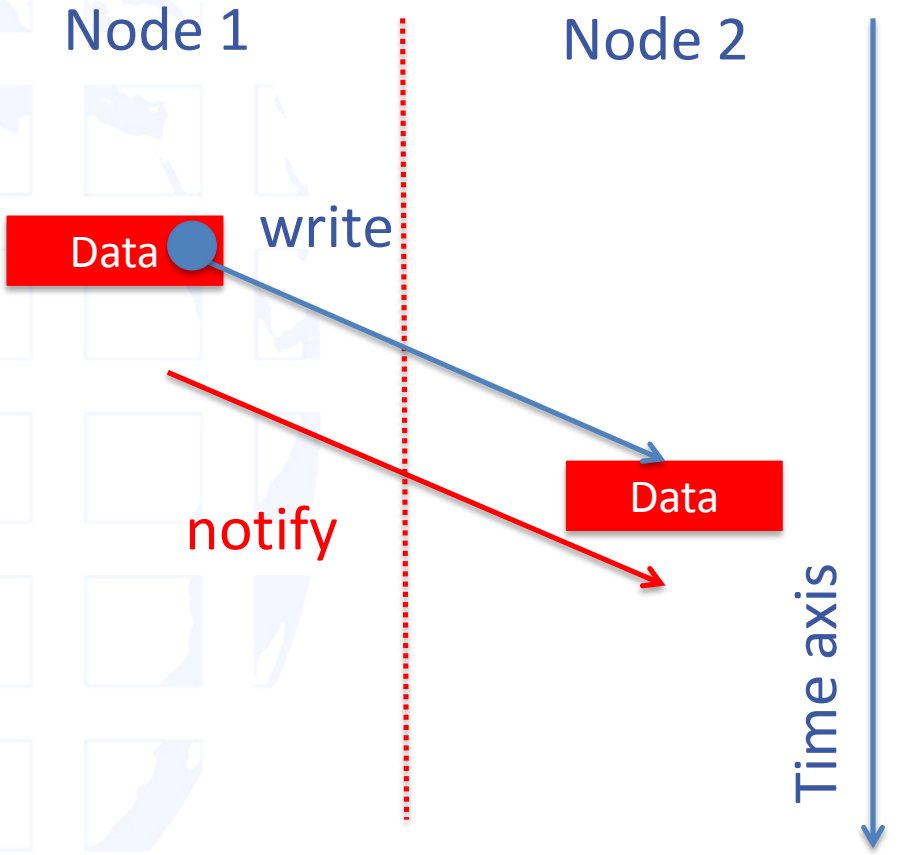
One-sided Communication

- One-sided operations between parallel processes include remote reads and writes.
- One-sided operations are not bound to a time - epoch.
- Asynchronous, (queue based) execution model.
- One-sided communication is non-blocking - communication is initiated but might not be finished yet.



Synchronisation with Notifications

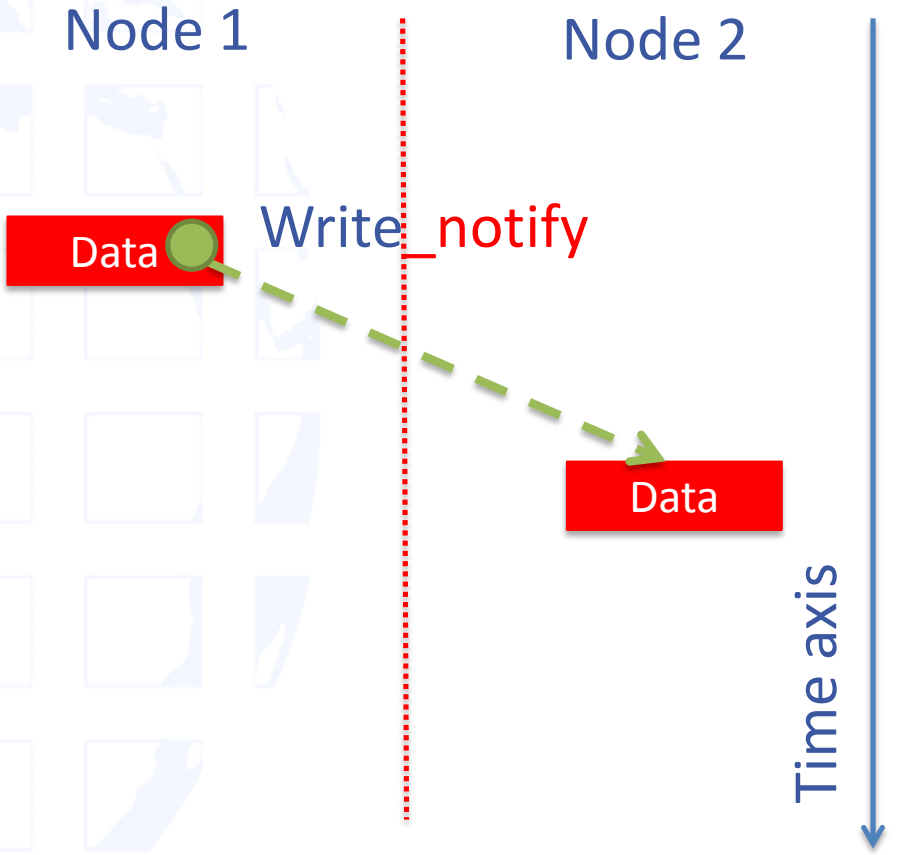
- We can issue mutiple writes.
- Node 2 has not participated.
- Node 2 does not know that communication has started
- Node 2 has to be **notified**.
- Node 2 can test or wait for notified communication.



Global
Address Space
Programming Interface
GASPI

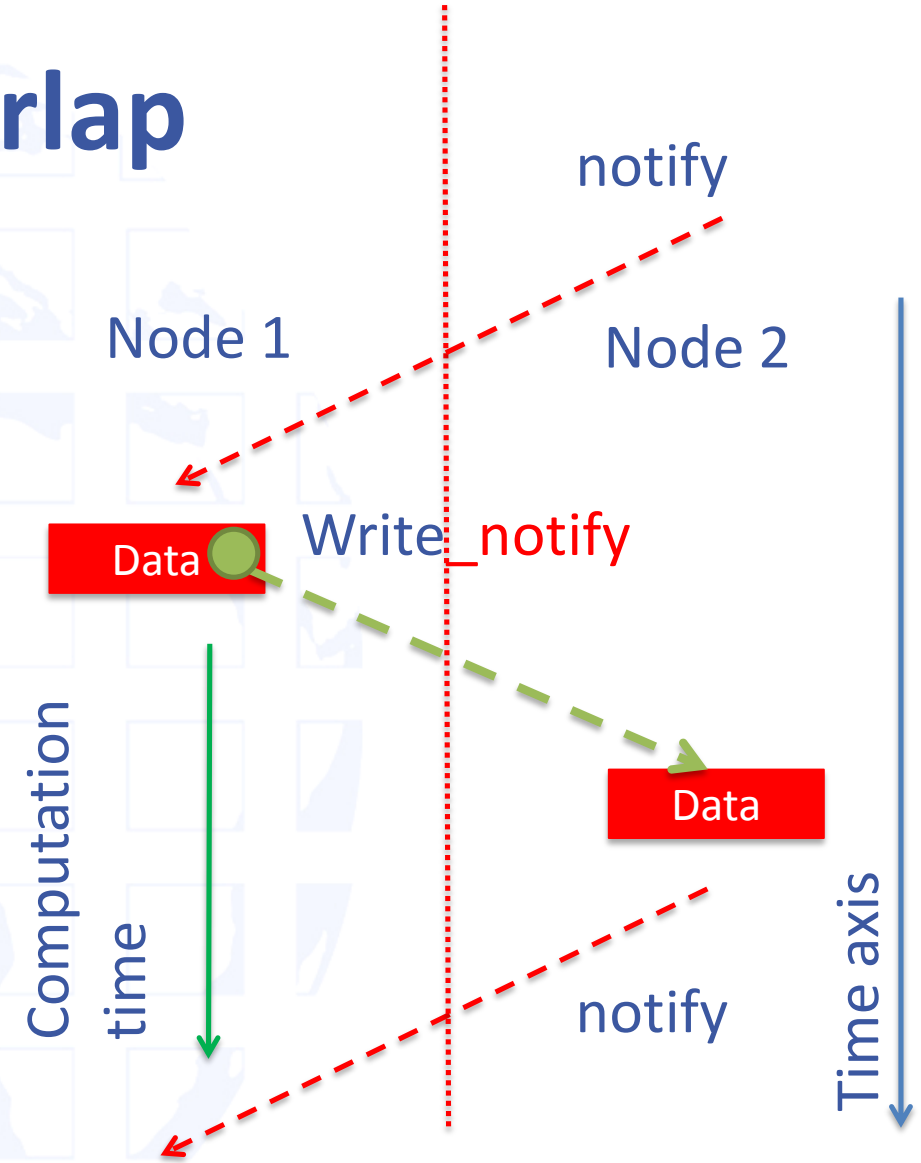
Synchronisation with Notifications

- GASPI supports a fused **write** and **notify** as a single call: write_notify.
- Today we have native support for write_notify in e.g. Cray Aries, Infiniband, Extoll.



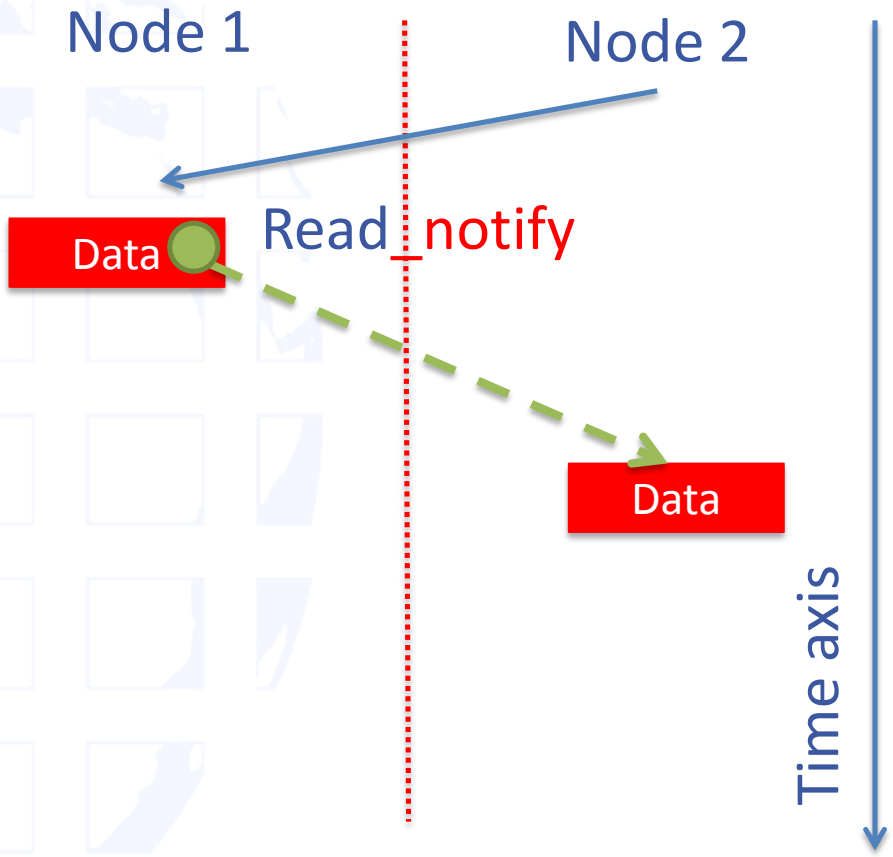
Overlap

- Communication and computation can happen in parallel
- Communication latency is hidden
- We could acknowledge the notify (or flag a request for write_notify).



Synchronisation with Notifications

- And the same for **read**:
  - Read_notify



Global
Address Space
Programming Interface
GASPI

Failure Tolerance

- Timeouts in all non-local operations
- Timeouts for Read, Write, Wait, Segment Creation, Passive Communication.
- Dynamic growth and shrinking of node set.
- Explicit connection management.
- Supports fast Checkpoint/Restart to NVRAM.
- State vectors for GASPI processes.

Interoperability with MPI

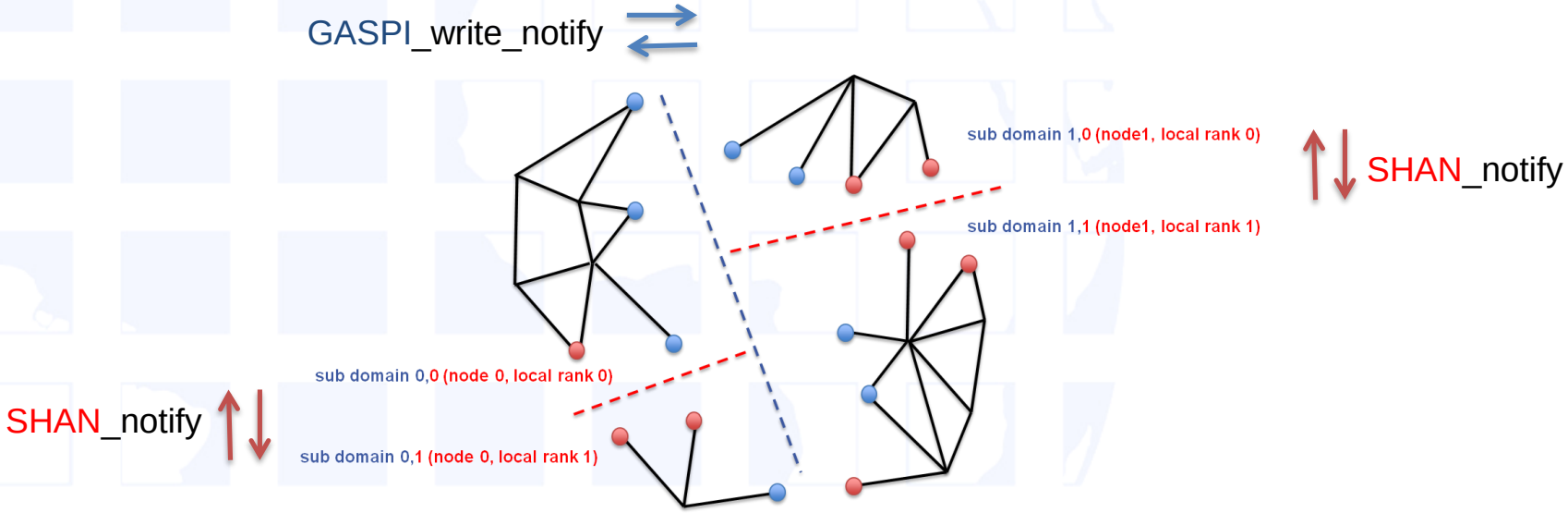
- GASPI supports **interoperability** with MPI
- The mixed-mode
  - Complements MPI with a (Partitioned) Global Address Space.
  - Allows replacing performance-critical parts of an MPI based application with GASPI code.
  - Inherits startup, rank id and number of ranks from MPI.
  - Caveat: No fault tolerance
- Porting guides available at:

  http://www.gpi-site.com/gpi2/docs/whitepapers/

Global
Address Space
Programming Interface
GASPI

GASPI-SHAN

- Shared GASPI windows: Migration for flat MPI legacy code.
- Domain decomposition: inter-node GASPI, inner-node SHAN.
- Data flow models.



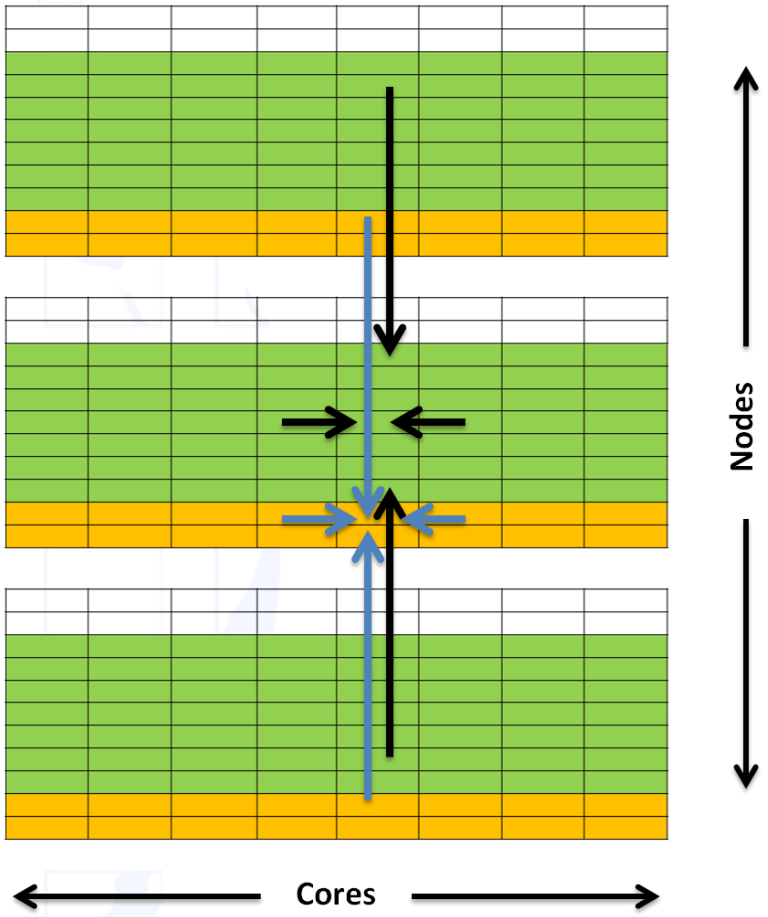
GASPI

- GASPI notifications:
  Notified communication
  One-sided, bundled.

Notified communication.
- Write: pack, one-sided write (**network**)

Black == Data

Blue == Notification (GASPI)



Global
Address Space
Programming Interface
GASPI

GASPI-SHAN

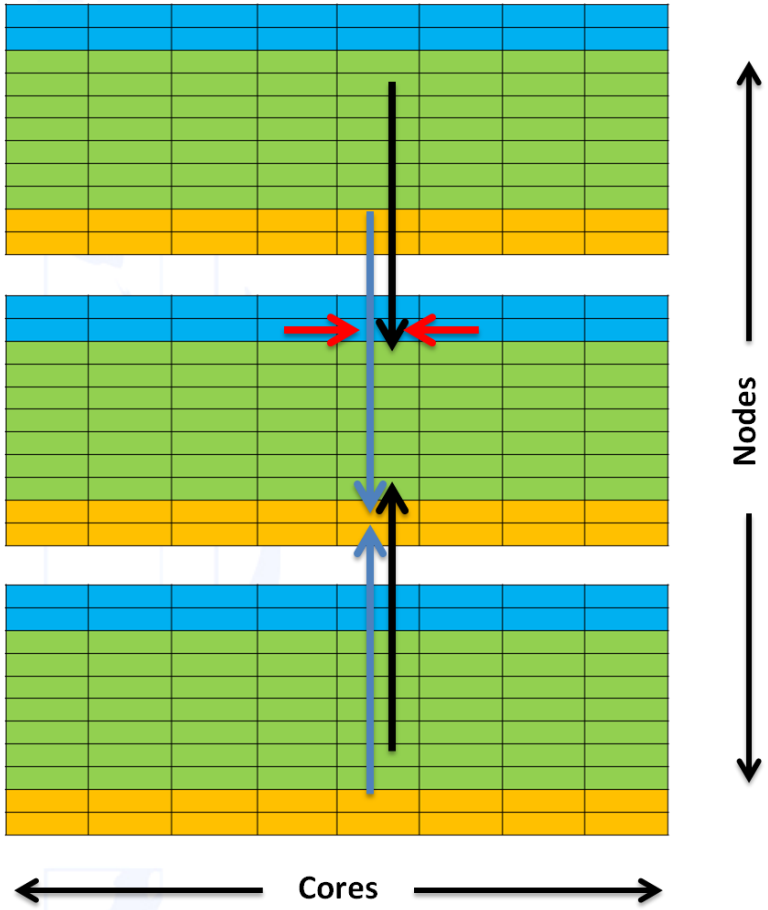
- SHAN notifications:
  Notified communication.
  One-sided, replaces write with "readable"
  Replaces wait for write with "wait-for-read"

Notified communication
- Notify or write

Black == Data

Blue == Notification (GASPI)

Red == Notification (SHAN)

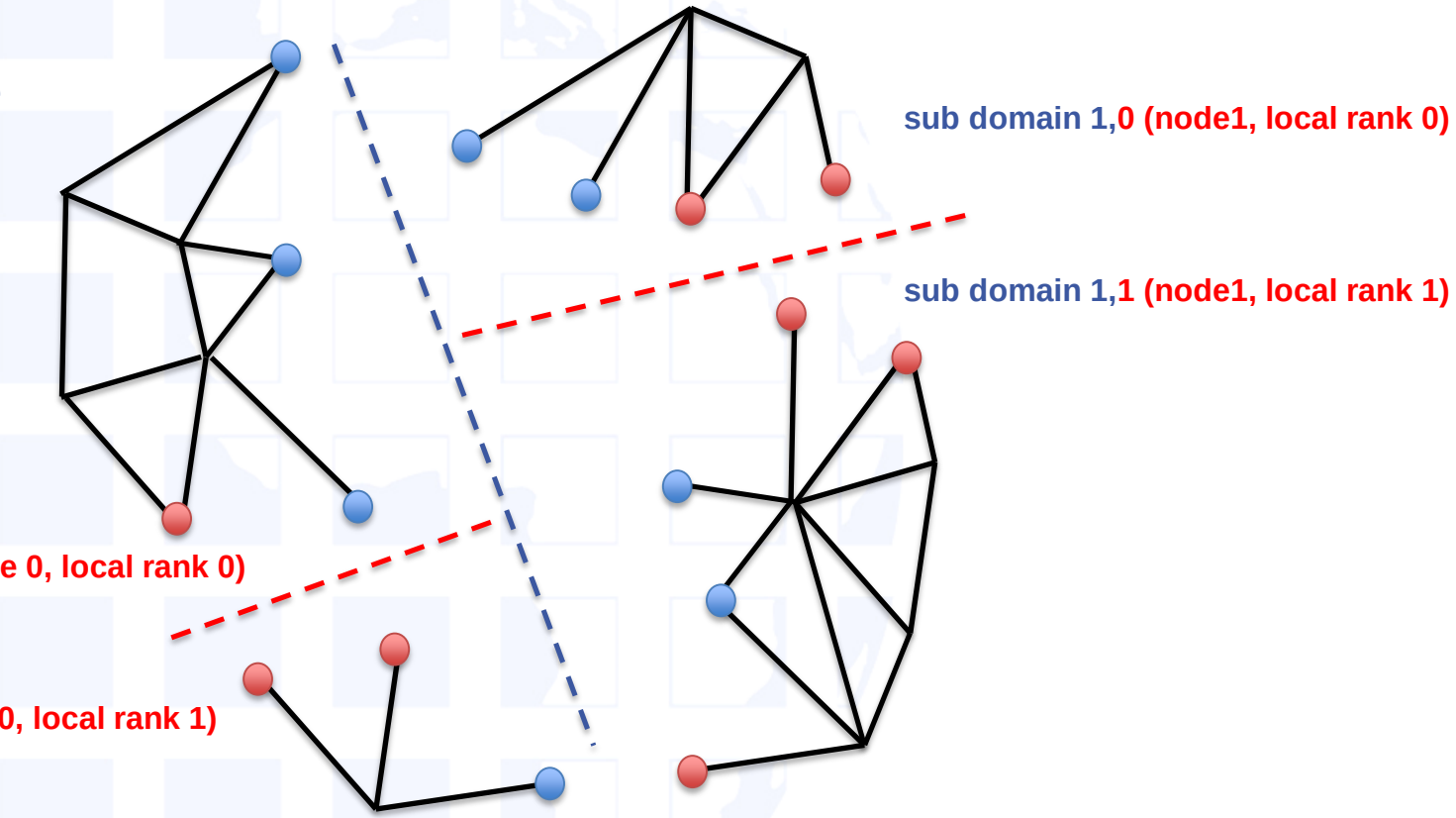


Global
Address Space
Programming Interface
GASPI

Ghost Cell Exchange

Publish ghost cell meta-data (types) and data across intra-node

- intra-node
- inter-node



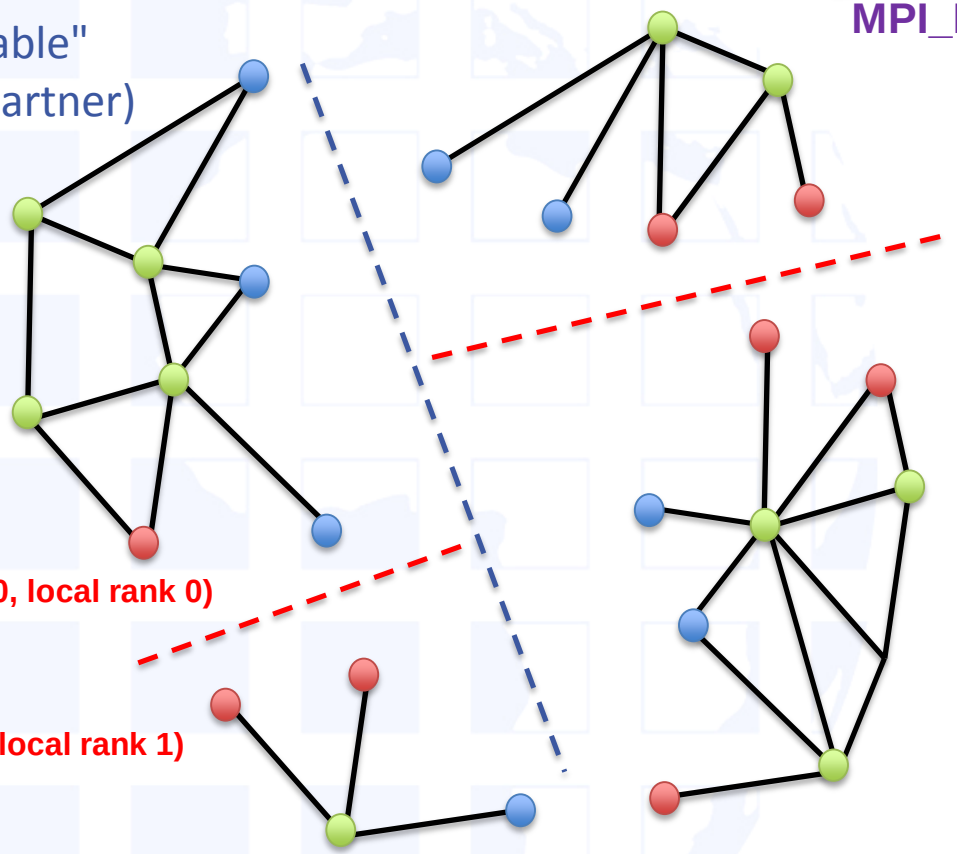
Global
Address Space
Programming Interface
GASPI

Ghost Cell Exchange

SHAN notify - intra-node

- Flag as "readable" (per comm_partner)

~~MPI_Irecv(.., recv_req)~~
MPI_Isend(..,send_req)

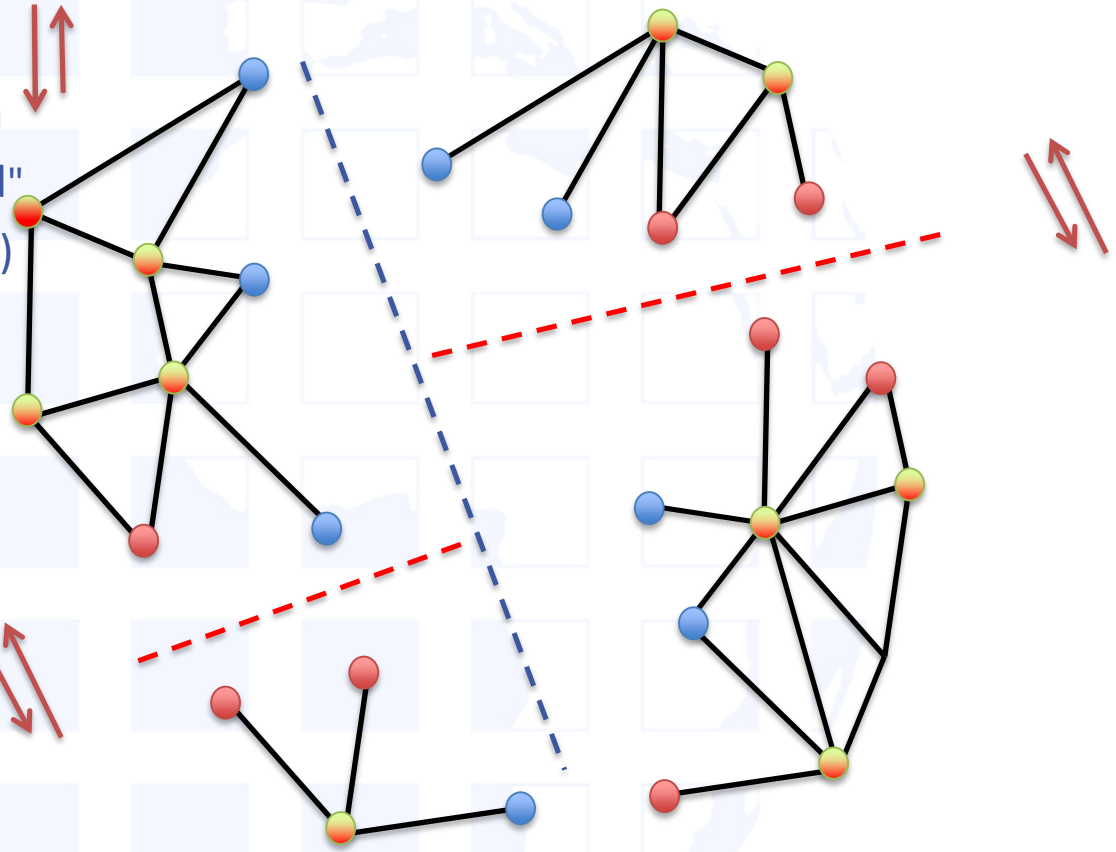


Ghost Cell Exchange

SHAN recv and ack - intra node

- Read intra-node data type and data
- Flag as "have_read" (per comm_partner)

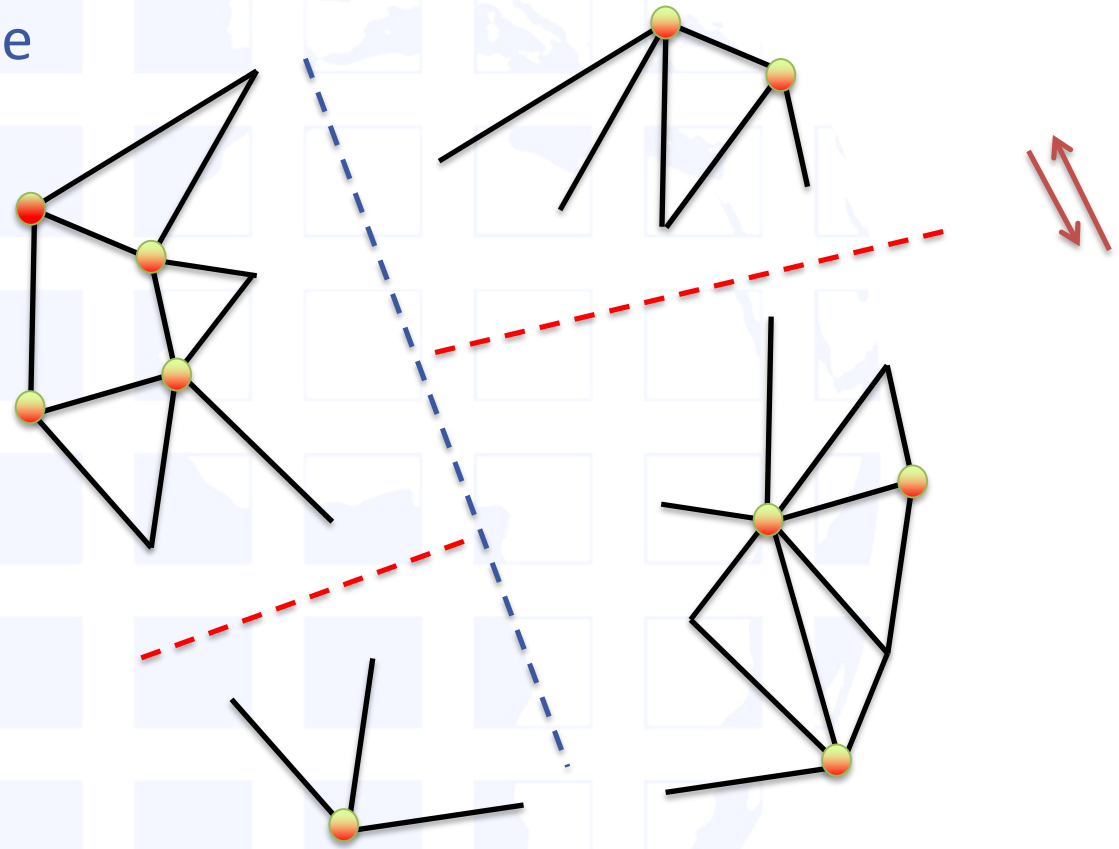
MPI_Waitany(.., recv_req)



Ghost Cell Exchange

SHAN wait for "have_read" (ack)
and continue

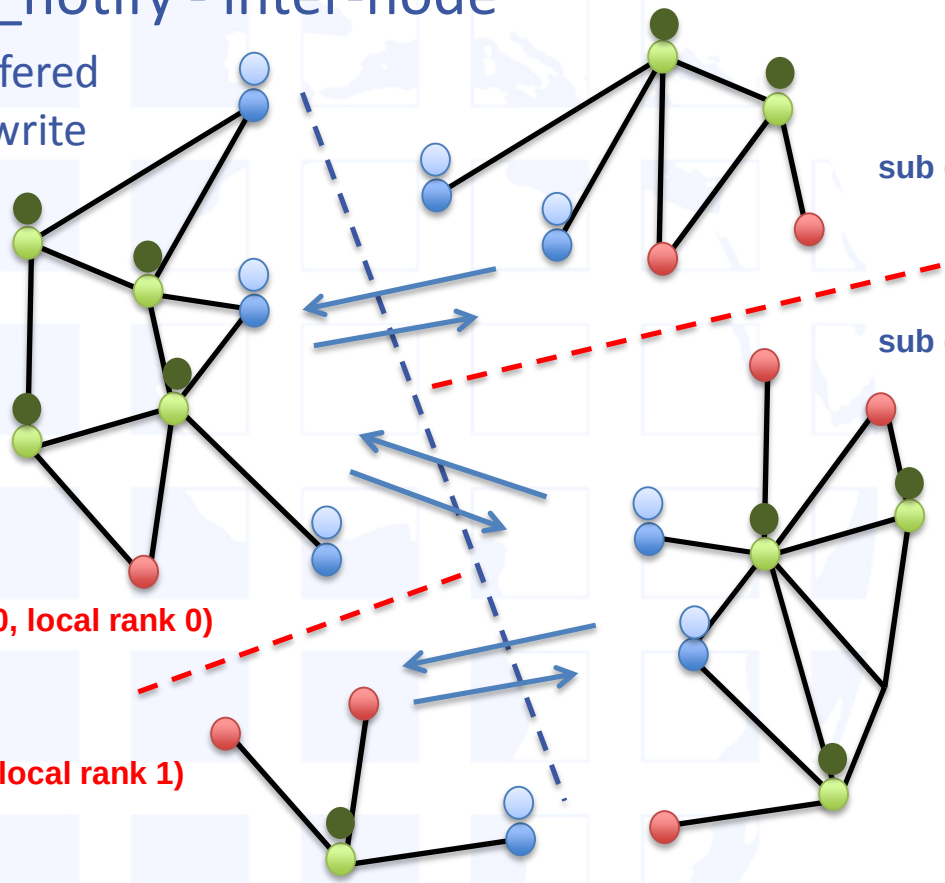
MPI_Waitany(.., send_req)



Ghost Cell Exchange

GASPI_write_notify - inter-node

double-buffered
notified write



sub domain 1,0 (node1, local rank 0)

sub domain 1,1 (node1, local rank 1)

sub domain 0,0 (node 0, local rank 0)

sub domain 0,1 (node 0, local rank 1)

Global
Address Space
Programming Interface
GASPI

Allocation

- Shared memory allocation for application data.

```
/** Local allocation of shared memory of size dataSz
 *
 * Note: Memory will be page-aligned.
 *
 * @param segment      - segment handle
 * @param shan_id      - (unique) segment id
 * @param shan_type    - type of allocated memory
 * @param dataSz       - required memory size per rank in byte
 * @param MPI_COMM_SHM - shared mem communicator
 *
 * @return SHAN_SUCCESS in case of success, SHAN_ERROR in case of
error.
 */
int shan_alloc_shared(shan_segment_t *const segment
                    , const int shan_id
                    , const int shan_type
                    , const long dataSz
                    , const MPI_Comm MPI_COMM_SHM
        );
```

```
/** Gets shared mem pointer for node local ranks
 *
 * @param segment      - segment handle
 * @param rank         - node local rank id
 * @param shm_ptr      - required memory size per rank in byte
 *
 * @return SHAN_SUCCESS in case of success, SHAN_ERROR in case of
error.
 */
int shan_get_shared_ptr(shan_segment_t * const segment
                      , const int rank
                      , void **shm_ptr
    );
```

Global
Address Space
Programming Interface
GASPI

Neighborhood

- Persistant communication in neighborhoods

```
/** Initialize persistant communication for shared mem and GASPI.
 *  requires bidirectional communication for synchronization in
 *  one-sided communication.
 *
 *  A zero length messages will work, no message at all will fail.
 *
 *  - allocates shared and private mem for communication
 *  - figures out local and remote comm partners.
 *  - negotiates remote number of neighbors and comm index
 *
 * @param neighborhood_id - general neighborhood handle
 * @param neighbor_hood_id - neighborhood id
 * @param neighbors      - comm partners (neighbors)
 * @param num_neighbors - num comm partners (neighbors)
 * @param maxSendSz      - max send size for every type.
 * @param maxRecvSz      - max recv size for every type
 * @param max_nelem_send - max number of send elements per type
 * @param max_nelem_recv - max number of recv elements per type
 * @param num_type       - number of types
 * @param MPI_COMM_SHM - MPI shared mem communicator
 * @param MPI_COMM_ALL - embedding of shared communicator
 * @return SHAN_SUCCESS in case of success, SHAN_ERROR in case of
error.
 */
int shan_comm_init_comm(shan_neighborhood_t *const neighborhood_id
                        , int neighbor_hood_id
                        , int *neighbors
                        , int num_neighbors
                        , long *maxSendSz
                        , long *maxRecvSz
                        , int *max_nelem_send
                        , int *max_nelem_recv
                        , int num_type
                        , MPI_Comm MPI_COMM_SHM
                        , MPI_Comm MPI_COMM_ALL
    );
```

Types

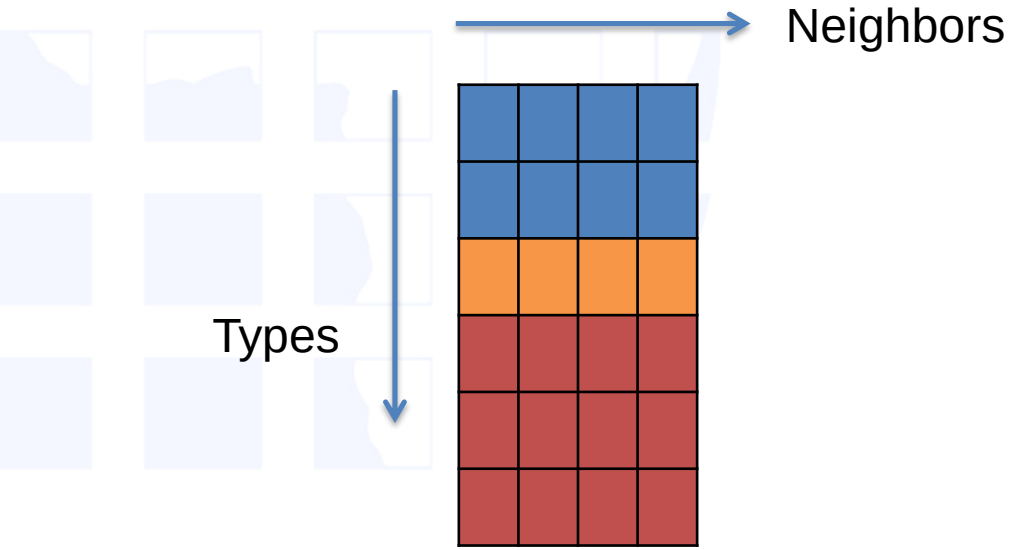
- Shared memory for data types: SHAN type description

```
/** Gets type data structure for node local ranks
*
 * @param neighborhood_id - general neighborhood handle
 * @param type_id         - used type id
 * @param nelem_send - pointer to number of send elements in shared
mem
 * @param nelem_recv - pointer to number of recv elements in shared
mem
 * @param send_sz         - pointer to send size in shared mem
 * @param recv_sz         - pointer to recv size in shared mem
 * @param send_offset     - pointer to offset of send elements in
shared mem
 * @param recv_offset     - pointer to offset of recv elements in
shared mem
 *
 * @return SHAN_COMM_SUCCESS in case of success, SHAN_COMM_ERROR in
case of error.
 */
int shan_comm_type_offset(shan_neighborhood_t *neighborhood_id
                          , int type_id
                          , int **nelem_send
                          , int **nelem_recv
                          , int **send_sz
                          , int **recv_sz
                          , long **send_offset
                          , long **recv_offset
    );
```

Global
Address Space
Programming Interface
GASPI

Types

- Send/Recv type offsets for individual neighbours are spaced apart by the maximal number of allocated max_nelem_send/recv elements per type.
- Offsets, Send/Recv sizes and currently used number of send/recv elements can be changed on the fly.
- Allocated ressources are dedicated and exclusive per type and per neighbor.



INTERTWinE

Global
Address Space
Programming Interface
GASPI

Communication

- SHAN communication: notify or write

```
/** Writes data or flags data as readable.
 *
 *  - aggregates send data into linear buffer or
 *  - flags data as readable
 *     - number of elements
 *     - element sizes and
 *     - element offsets
 *
 * @param neighborhood_id - general neighborhood handle
 * @param data_segment    - data segment handle
 * @param type_id         - type index
 * @param idx             - comm index for target rank in neighborhood
 *
 * @return SHAN_COMM_SUCCESS in case of success, SHAN_COMM_ERROR in
case of error.
 */
int
shan_comm_notify_or_write(shan_neighborhood_t *const neighborhood_id
                          , shan_segment_t *data_segment
                          , int type_id
                          , int idx
    );
```

Global
Address Space
Programming Interface
GASPI

Communication

- SHAN communication: test or wait

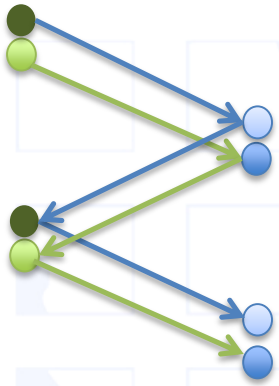
```
/** Tests for specific receive requests.
 *
 * @param neighborhood_id - general neighborhood handle
 * @param data_segment    - data segment handle
 * @param type_id         - type index
 * @param idx             - comm index for target rank in neighborhood
 *
 * @return SHAN_COMM_SUCCESS in case of success, SHAN_COMM_ERROR in case of error.
 */
int shan_comm_test4Recv(shan_neighborhood_t *const neighborhood_id
                        , shan_segment_t *data_segment
                        , int type_id
                        , int idx
    );
```

```
/** Tests for specific send requests
 *
 * @param neighborhood_id - general neighborhood handle
 * @param type_id         - type index
 * @param idx             - comm index for target rank in neighborhood
 *
 * @return SHAN_COMM_SUCCESS in case of success, SHAN_COMM_ERROR in case of error.
 */
int shan_comm_test4Send(shan_neighborhood_t *const neighborhood_id
                        , int type_id
                        , int idx
    );
```

Ghost Cell Exchange

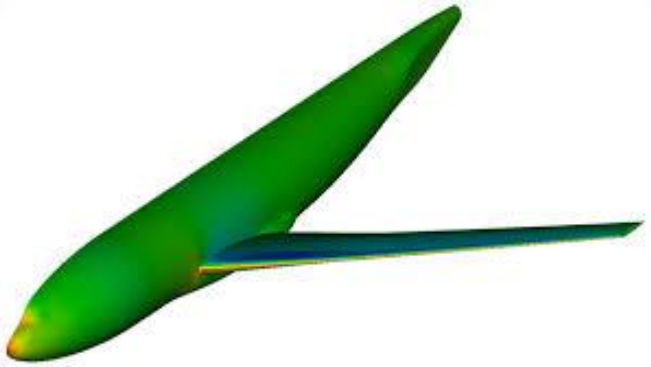
A brief note on double buffered notified communication:

- Bi-directional double buffered notified writes (or reads) will avoid potential data races.



Global
Address Space
Programming Interface
GASPI

Hands-On



SHAN

https://gitlab.com/csimmend/GASPI-SHAN.git

Hands-On: CFD-Proxy

Code migration from flat MPI towards notified communication.

Challenge:

Set the required meta data information for nelem_send nelem_recv, send_sz, recv_sz, send_offset, recv_offset.

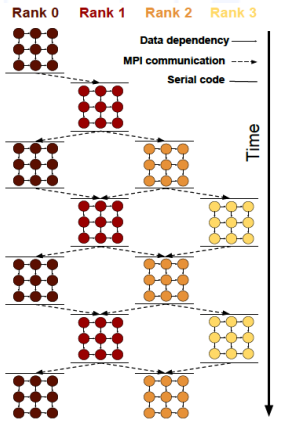
Hands-On

Hands-On: Heat

Code migration from flat MPI towards data dependency driven execution.

Challenge:

Set the required data dependencies for the stage counters for sendFirst, recvUpper, recvLower, solve, sendLast, testLast. (011.solver_gaspi_shan.cpp)



Global
Address Space
Programming Interface
GASPI

Questions?

Thank you for your attention

www.gaspi.de