RWTH AACHEN UNIVERSITY

MPI Caveats

When correct MPI programs did not run

Paul Kapinos[1]
Hans Rüdiger Hammer, Alexander Aures[2]

(1) kapinos@rz.rwth-aachen.de
(2) [hammer|aures]@lrst.rwth-aachen.de

Rechen- und Kommunikationszentrum (RZ)

MPI = Message Passing Interface

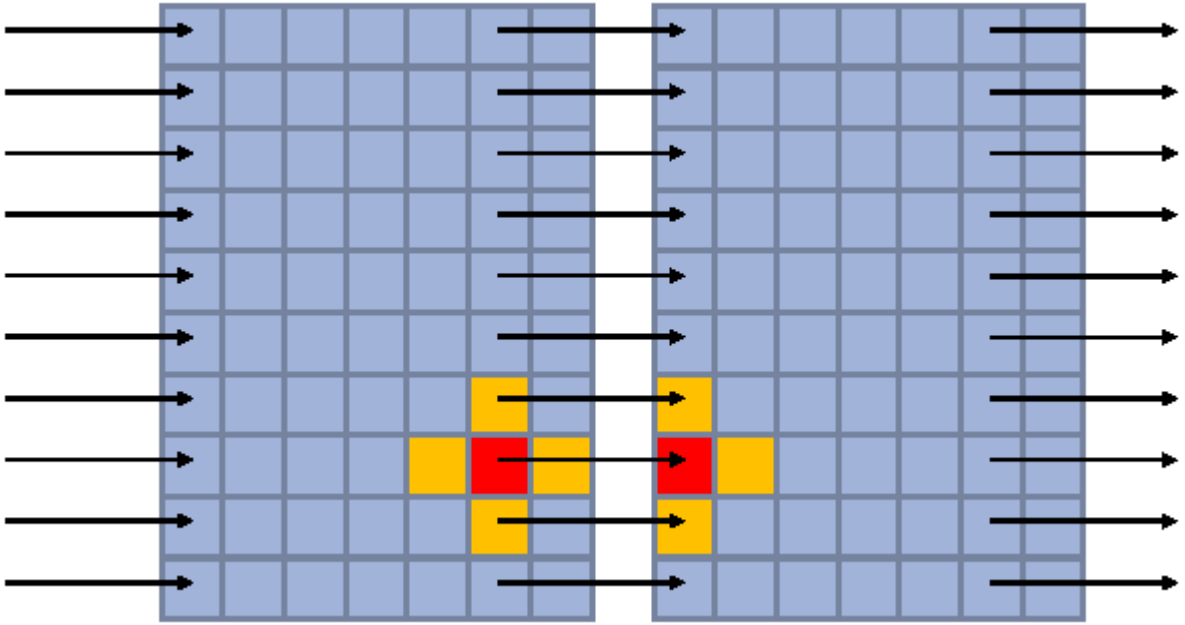
- **The portable way to program distributed memory computers [1]**
- ***"… you call MPI_SEND. A miracle occurs. The message is received on the other side. …"* Jeff Squyres, [2]**
- **A correct MPI program will run in any environment (regardless which library vendor, hardware, operating system) supporting the same standard**
- **… if the MPI library, the OS, the hardware did not bring own limitations**

The World has changed

- **typical compute node has...**
  - → 1992: 1 CPU, a handful of MegaBytes RAM
  - → 2012: 12+ cores, 24-96 GigaByte RAM
  - → (a real node @RZ: 128 cores, 2 TeraByte RAM)
- **memory footprint raised in order of 1000+**
- **network speed enhanced accompanying**
  - → (10Mbit/s (Ethernet) => 40Gbit/s (QDR InfiniBand)

Typical MPI program

- 2D Domain decomposition, only tiny „halo“ has to be exchanged



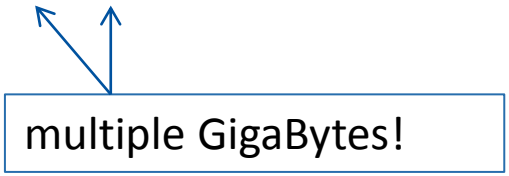
- Data: $O(N^2)$, halo: $O(N)$
- The volume of messages is „small“

Not that typical MPI program

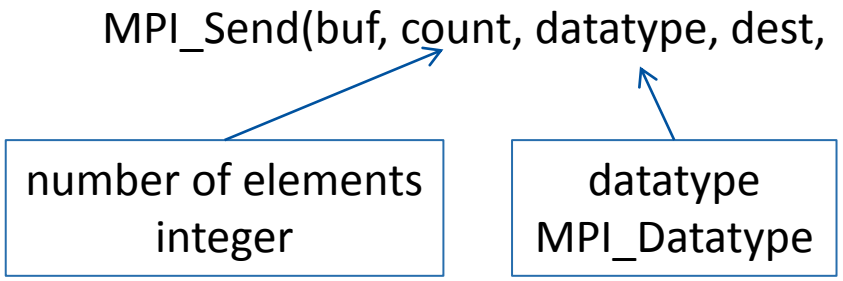
- **There are also MPI programs transferring *a lot* of data (full arrays).**

- **... Serpent is one of such programs!**

```
MPI_Reduce(dat, buf, sz, MPI_DOUBLE, MPI_SUM, root, MPI_COMM_WORLD);
```



The frontiers in the MPI standard



MPI_Datatype	C datatype	Size, bytes
MPI_DOUBLE	double	8
….		
MPI_BYTE	-	1

- **Current computers usually use the LP64 model [3]**
- **LP64: Integer data type still 32 bits (ILP64: integer 64 bits)**
  - → Max(int) = 2^{31} = 2,147,483,648 ≈ $2*10^9$
  - → Max. Message Size (array of doubles) ≈ 16 GigaByte
- **Not *that* much compared to 16 GB/core on actual computers, huh?**
- **Yes you may get round this by defining own data types:**
  - → create a datatype comprised of 1024 contiguous doubles,
  - → send 2^{31} of those ≈ 16 TeraBytes (that's will be enough for next days)

The frontiers in the MPI implementations

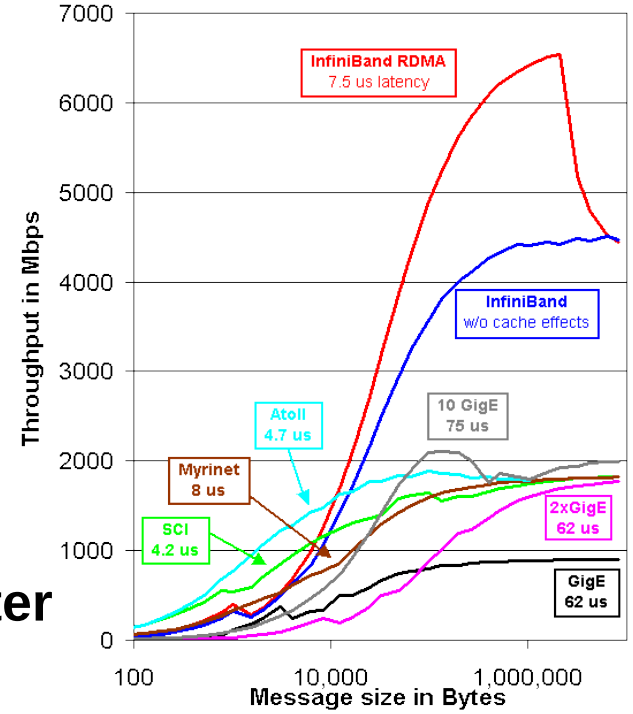
- ***Every* underlying networking deals in numbers of bytes…**
- **Intel MPI: max. message size (any data type) ≈ 2 GigaByte**
  - → probably compute internally the number of *bytes* to be transferred and store it in an 32bit integer.
- **Note: change LP64 -> ILP64 (32->64bit integers) usually bumpy [3]**
  - → esp. in close-to-metal programming
- **Dear programmer, transfer big data in chunks!**
- **Serpent already does this ☺**

Future solution in Intel-MPI

- **“The current limitation of the 2GB maximum message size is due to the type ‘int’ used by the MPI standard to designate a message size parameter. On the LP64 programming model, which is used overwhelmingly in engineering software, int is 32 bits long, which drives the limitation. The development team has decided to stay with the LP64 model at this time. They will be improving the error message displayed when the message size is beyond the limitation.” James Tullos**

Send it in chunks

- **Max. message size as mentioned (a few GigaBytes)**
- **The performance is usually bad for extremely huge messages**
  - → we see 5% loss (PingPong test, >8MB/msg)
  - → also see http://www.scl.ameslab.gov/netpipe/
- **The same for small messages**
  - → 100 Mb/chunk is too much
  - → 10 Kb/chunk is too small
  - → we would recommend 1-8 Mb/chunk
- **Ideally the chunk size is a runtime parameter**
  - → allow fine tuning for actual system



- **usually, the MPI library can be tuned for certain message sizes (e.g. for [4] Open MPI), but don't overoptimise your system!**

What to do if no access to source code?

- **There is still a way to split large messages**
  - → Use MPI profiling interface (PMPI) to intercept calls to MPI routines
  - → Write wrapper(s) for splitting large data in chunks
  - → build a library containing the wrapper(s)
  - → plant this library to the EXE using LD_PRELOAD envvar
- **Overhead?**
  - → splitting the message, intercepting the calls over PMPI interface
- **MPI_Bcast in 1MB chunks:**
  - → Intel MPI, 2GB : 12% loss
  - → (Open MPI, 2-8GB: less than 1% loss, 16GB: 4x slower!)
- **But may be the only way to run an actual data set…**

The frontiers of hardware / OS

- **Autumn 2011, a problem with Serpent (+ Open MPI 1.5.3):**
- ***"It stops working".*** **Huh.**
- **(… long investigations …)**

- **In one of the subsequent calls to MPI_Reduce (chunks!), the Serpent hung up, if using Open MPI over InfiniBand**

  → but run over Ethernet and with Intel MPI

- **How to interpret this?**

- **first guess was: "Bug in Open MPI"**
- **(… off-line and on-line discussions, [5])**

- **But it was much more thrilling!**

Registered memory, RDMA

- **Open MPI use Remote Direct Memory Access (RDMA). [11]**
- **perform *fast* message passing over InfiniBand**
- **For RDMA, memory must be *registered.***

  → the mapping of virtual pages to physical memory fixed (OS kernel's task)

- **There are parameters in the OS driver of the InfiniBand card (MTT parameters) which affect how many memory can be registered! [7]**

  → *log_num_mtt* (old hardware: *num_mtt*) (our default: 20)

  → *log_mtts_per_seg* (our default: 3)

  → max_reg_mem = $(2^{log\_num\_mtt}) * (2^{log\_mtts\_per\_seq}) * (4\ kB)$ (our default: 32 GB)

- **If there is less registered memory than needed, the behaviour is implementation defined.**

Too little registered memory 1

- **Open MPI 1.5.x (and probably older):**
  - → just hang up
  - → block an InfiniBand interface
  - → *another* jobs trying to communicate over this interface print out:

```
……..
--------------------------------------------------------------------------
WARNING: There was an error initializing an OpenFabrics device.

  Local host:   linuxbmc1008.rz.RWTH-Aachen.DE
  Local device: mlx4_0
--------------------------------------------------------------------------
```

Too little registered memory 2

- **Open MPI 1.6.x:**
  - → try to workaround this
  - → print a more (1.6.1) or less (1.6) meaningful warning

```
--------------------------------------------------------------------------
WARNING: It appears that your OpenFabrics subsystem is configured to only
allow registering part of your physical memory.  This can cause MPI jobs to
run with erratic performance, hang, and/or crash.
……
   http://www.open-mpi.org/faq/?category=openfabrics#ib-locked-pages
  Local host:           linuxbmc0013.rz.RWTH-Aachen.DE
  Registerable memory:     32768 MiB
  Total memory:         98293 MiB
--------------------------------------------------------------------------
```

- **There are still cases when the workaround did not catch and jobs die.**
  - → e.g. we have seen a problem with a (synthetic) example [8]

Too little registered memory 3

- **Intel MPI (4.x) often runs O.K.**
  - → it use InfiniBand not directly but over DAPL [10]
  - → DAPL mask/workaround the lack of Registered Memory
  - → (also Serpent run O.K. using Intel MPI)
- **but it also has issues rooted in lack of registered memory**

```
... ...
[81:linuxbdc04][../../dapl_conn_rc.c:128] error(0x30000): ofa-v2-ib0: could
not create DAPL endpoint: DAT_INSUFFICIENT_RESOURCES()
Assertion failed in file ../../dapl_conn_rc.c at line 128: 0
internal ABORT - process 81
linuxbdc04.rz.RWTH-Aachen.DE:272d:a6ad1720: 10854191 us(10854191
us!!!): reg_mr Cannot allocate memory
linuxbdc04.rz.RWTH-Aachen.DE:272a:7ed0f720: 10855903 us(10855903
us!!!): reg_mr Cannot allocate memory
……
```

Registered memory – what is to be done?

- **Use the newest version of MPI library!**
- **Set the MTT parameters to some suitable value! [7]**
  - → Mellanox and Open MPI say “2x RAM must be registrable”
  - → both say *“adjust* log_num_mtt*, not* log_mtts_per_seg*”*
- **What did the parameter do? See post of Yevgeny Kliteynik [9]**
  - → *“log_num_mtt* controls number of MTT segments,
  - → *log_mtts_per_seg* controls number of entries per segment.” -YK
  - → if *log_mtts_per_seg* set to a high value, fragmentation may became a problem (references to the same registration may be saved in a segment)
- **So, we join: raise *log_num_mtt* to get some *2x RAM* size registrable**

	log_num_mtt	*log_mtts_per_seg*	max_reg_mem
our default	20	3	$(2^{20}) * (2^3) * (4\ kB) = 32\ GB$
our choise	22	3	$(2^{22}) * (2^3) * (4\ kB) = 128\ GB$

What about the performance?

- **We did not see any performance change by going from default (20/3) to 22/3 on both node types (with 24GB and 96GB RAM)**
  - → Linpack (6 nodes, 72 processes, 12 ppn)
  - → bandwidth measurements (6 nodes, 6 processes, 1 ppn)
  - → nevertheless, we recommend to test *your* application in *your* environment
- **Don’t raise the parameters over the hills and far away**
  - → 23/6 leads to unreachable InfiniBand interfaces,
  - → whereas both 23/3 and 20/6 have worked fine.

Overview

- **Correct MPI programs may still fail due to problems in the MPI library, OS, hardware.**
- **@Programmer: Don’t send GigaBytes in a single piece, use chunks!**
- **@Admin/User: Check (and adjust) the MTT parameters!**
- **Are that all errors? Probably, not.**
  - → E.g. [6]: VASP (computational chemistry, http://www.vasp.at/) jobs produce curious behaviour of Linux (khugepaged run amok)
- **so @ALL, stay tuned!**

Thanks for the attention!

Would you like to know more?

- 1. http://www.mcs.anl.gov/research/projects/mpi/
- 2. http://www.aosabook.org/en/openmpi.html
- 3. http://www.unix.org/version2/whatsnew/lp64_wp.html
- 4. http://www.open-mpi.org/faq/?category=openfabrics#large-message-tuning-1.3
- 5. http://www.open-mpi.org/community/lists/users/2012/02/18565.php
- 6.http://www.open-mpi.org/community/lists/users/2012/08/20057.php
- 7. http://www.open-mpi.org/faq/?category=openfabrics#ib-low-reg-mem
- 8. http://www.open-mpi.org/community/lists/devel/2012/08/11396.php
- 9. http://www.open-mpi.org/community/lists/devel/2012/08/11417.php
- 10. http://www.datcollaborative.org/
- 11. http://en.wikipedia.org/wiki/Remote_direct_memory_access