Improving Gaussian’s parallel performance using Infiniband.

Authors: Roy Dragseth, roy.dragseth@uit.no
Espen Tangen, espen.tangen@uit.no
The IT-Department, University of Tromsø

This work has been conducted as a part of the Performance Analysis Workgroup within the Norwegian Consortium for High Performance Computing (www.notur.no).

Abstract

The Gaussian™ computational chemistry application is widely used by researchers in the field of theoretical chemistry and consumes a large amount of CPU-cycles on HPC-centers around the world.

By employing a standard system trick it is possible to dramatically improve the parallel performance of the Gaussian application on compute clusters with Infiniband networks. This can be achieved without changing the application itself.

Background

The Gaussian application[1] is using thread parallelism within a compute node and TCP-Linda[2] for communication between nodes. TCP-Linda runs over standard Ethernet using sockets limiting the possibility to achieve the same parallel performance as MPI applications that can easily be run over Infiniband networks. However, the OFED Infiniband software stack contains a wrapper library, Rsockets, giving the possibility to run socket based applications with near optimal speed over Infiniband networks. The wrapper library method can in principle be used on any systems with dynamic linking.

Benchmark results

One test molecule was chosen and two different computational methods were employed to demonstrate the improvement in performance using Infiniband instead of Ethernet as the communication layer. The computational methods used were

1. hybrid functional B3LYP
2. pure functional BP86

All runs were performed on the same set of compute nodes using three different network technologies.

[1] Gaussian is a trademark of Gaussian Inc, http://www.gaussian.com

[2] TCP-Linda is a trademark of Scientific Computing Associates Inc, http://www.lindaspaces.com

1. standard Ethernet (TCP)
2. IP over Infiniband (IPib)
3. Rsockets over Infiniband (IBr)

All benchmark runs used all 16 CPU-cores on each compute node.

Hybrid functional B3LYP

	TCP	Speedup	IPib	Speedup	IBr	Speedup
1 node	07:00:15		07:00:15		07:00:04	
2 nodes	04:01:23	1,74	03:52:14	1,81	03:45:06	1,87
4 nodes	03:12:16	2,19	02:30:35	2,79	02:05:04	3,36
8 nodes	02:50:14	2,47	02:01:08	3,47	01:17:16	5,44
16 nodes	02:54:28	2,41	02:03:21	3,41	01:04:49	6,48

Table 1 Walltime for hybrid method, (HH:MM:SS)

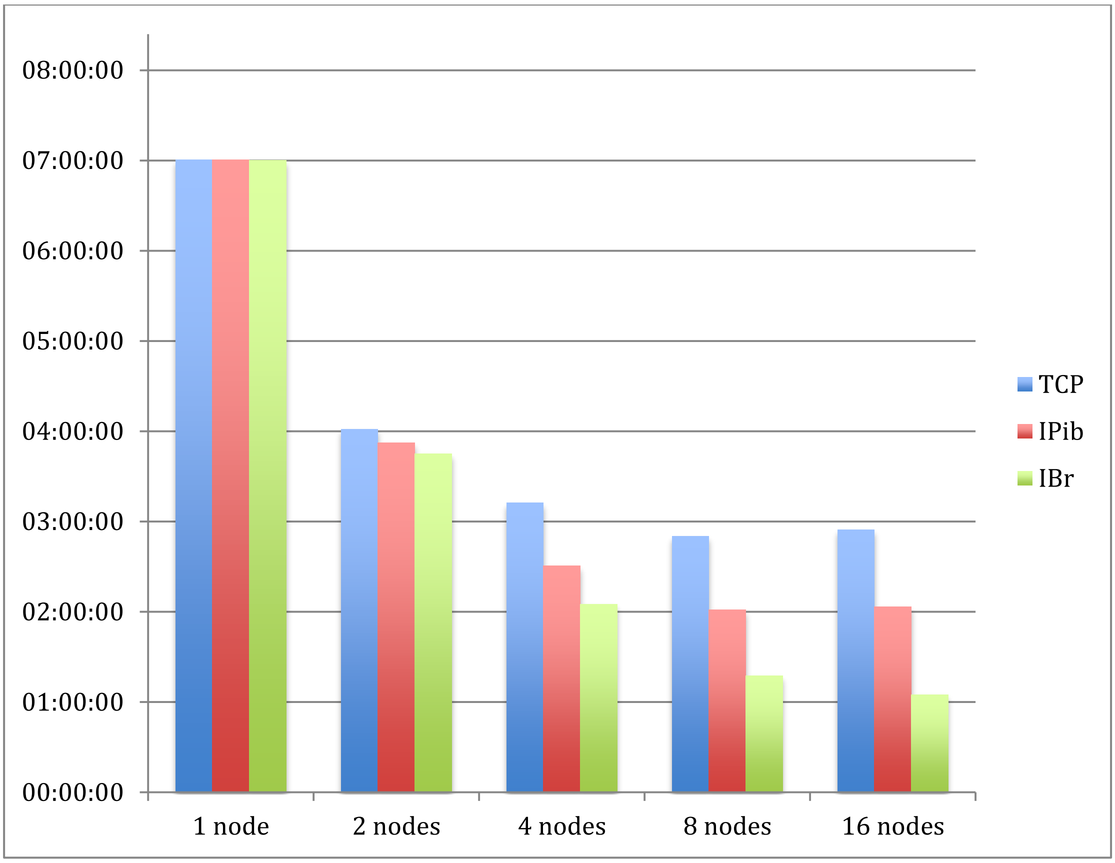


Figure 1 Performance chart for hybrid method. Walltime for each run , lower is better.

Pure functional BP86

	TCP	Speedup	IPib	Speedup	IBr	Speedup
1 node	03:36:07		03:36:17		03:36:37	
2 nodes	02:21:37	1,53	02:15:11	1,60	02:02:34	1,50
4 nodes	02:44:17	1,32	01:55:19	1,88	01:14:42	2,25
8 nodes	02:35:27	1,39	01:48:18	2,00	00:55:12	3,38
16 nodes	02:42:36	1,33	01:56:15	1,86	00:56:29	3,83

Table 2 Walltime for the pure functional method (HH:MM:SS)

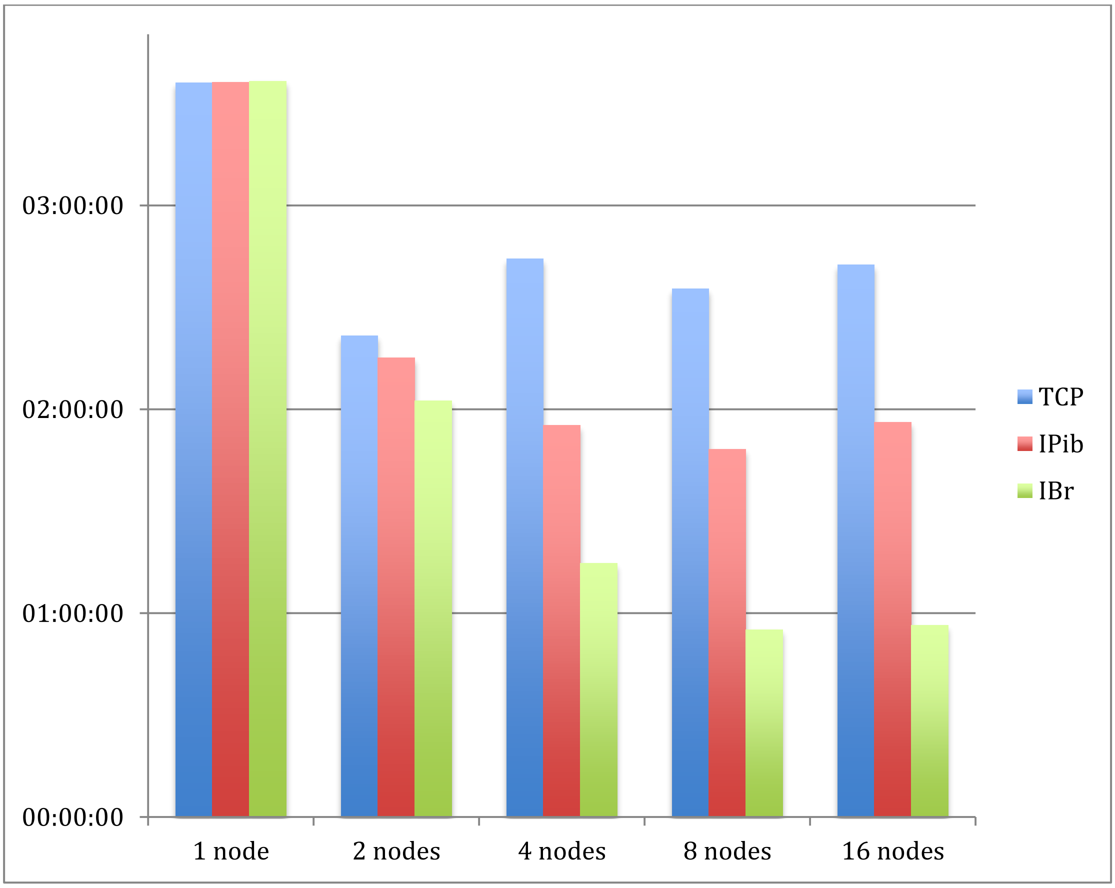


Figure 2 Performance results for the pure functional method. Lower is better.

Conclusions

The performance improvements gained by scaling out to increase parallelism using Rsockets over Infiniband is significantly higher than what can be achieved using regular Ethernet networks. The pure functional method does not scale beyond 2 nodes over Ethernet, but using Rsockets one achieves reasonable scaling even at 8 nodes. The hybrid method shows better scaling properties and gives > 5X speed improvement going from 1 to 8 compute nodes. Scaling beyond 8 nodes does not give any further performance improvement. The benchmark tests also show some performance improvement by using the IP-over-IB functionality, but real gain is only achievable using Rsockets.

Although perfect scaling and massive parallelism is not achievable for the Gaussian application the method of using Rsockets as the communication layer gives significant savings and improved efficiency for a wide range of users within the field of computational chemistry.

Technical description of the benchmark setup.

All benchmark tests were performed on the Stallo cluster (stallo.uit.no) using the same set of compute nodes for the different tests. More information about the Stallo system can be found at http://docs.notur.no/uit.

All input files and shell scripts can be downloaded from https://depot.uit.no/projects/g09overib/files

Hardware details.

Compute nodes: HP BL460c gen8 with two INTEL Xeon E2670 CPUs (8 CPUcores, 2.6GHz) and 32GB memory.

Ethernet network: 1 gigabit/second. Switch: HP GbE2c Layer 2/3 Ethernet Blade Switch.

Infiniband network: Mellanox QDR NICs. Switch: HP BLc 4X QDR IB Switch

In the Stallo system each blade enclosure contains one Ethernet and one Infiniband switch connecting the 16 nodes within the enclosure with each other. The Ethernet switch only have one 1 gigabit/second uplink shared by the 16 nodes in the enclosure, therefore all benchmarks were performed within one blade enclosure to prevent artificial performance differences when comparing Ethernet and Infiniband runs.

System software.

The Stallo system runs Rocks v6.0 and use CentOS 6.4 as the base system distribution. The Linux kernel version at the time of the benchmark runs were 2.6.32-279.2.1.el6.x86_64 as provided by the stock CentOS distribution. The Rsockets version was included in the librdmacm-1.0.17-0.git4b5c1aa.el6.x86_64 rpm as provided by CentOS 6.4.

Application software.

Gaussian 09.c01 binary version as provided by Gaussian Inc.

Benchmark input files.

For this study, we used a rather typical problem set – a molecule representing a typical job type (geometry optimization using DFT) and a medium sized problem (56 atoms, one transition metal, closed electronic shell molecule).

We chose to run the problem both using the hybrid functional B3LYP and the pure functional BP86, to also investigate the typical higher cost of hybrid calculations related to pure functional calculations.

For the B3LYP jobs, we used the following keywords in the G09 inputfile:

```
#p b3lyp/6-311+G(d,p) opt
```

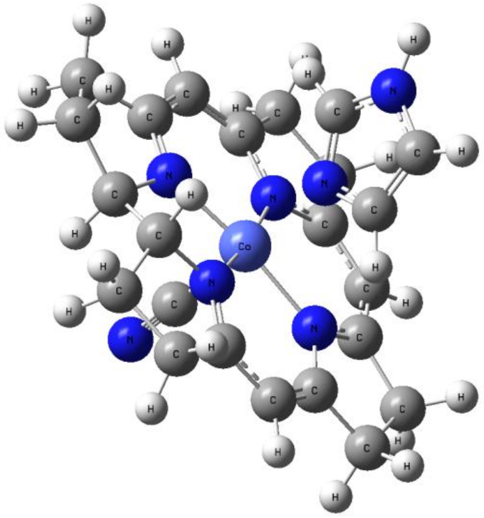
For the BP86 jobs, we used the following keywords in the G09 inputfile:

```
#p bp86/6-311+G(d,p) opt
```

The molecule was defined in the following way:

```
1 1
Co   -0.0074870    0.0192340   -0.2716695
C     0.0335883    0.1683795   -2.1095876
N     0.0742873    0.2577670   -3.2862856
N    -0.1758531   -0.9113487    3.8317795
N    -0.0866492   -0.1205425    1.7695227
C    -0.1438835   -1.2321558    2.5084140
C    -0.1351712    0.4676866    3.9488203
C    -0.0787617    0.9451498    2.6575989
H    -0.1621949   -2.2498409    2.1252021
H    -0.2224575   -1.5758707    4.6025010
H    -0.1515482    0.9693022    4.9126049
H    -0.0344760    1.9734880    2.3080337
N     1.4874938    1.1559102   -0.0981397
N    -1.2173614    1.5348571   -0.2556089
N    -1.4082350   -1.3175611   -0.3926574
N     1.3223269   -1.3005378   -0.4138257
C     2.8080390    0.5014281    0.0921212
C     3.8158995    1.5721709   -0.3722560
C     3.0515133    2.9004478   -0.1353880
C     1.5992603    2.4623427   -0.2061441
C     0.4716737    3.3212352   -0.3401017
C    -0.8284146    2.8646196   -0.3640259
C    -2.0389614    3.7609619   -0.5452945
C    -3.2295440    2.8037522   -0.3426211
C    -2.5717024    1.4392334   -0.3151168
C    -3.2897744    0.2432860   -0.3628773
C    -2.7384164   -1.0364375   -0.4239452
C    -3.5770119   -2.2917542   -0.5535154
C    -2.5324815   -3.4251390   -0.5230357
C    -1.2059116   -2.6887652   -0.4562674
C     0.0219837   -3.3199847   -0.4513398
C     1.2582781   -2.6131234   -0.4759762
C     2.6420839   -3.2183875   -0.6439881
C     3.5752262   -2.0240062   -0.3114729
C     2.6994301   -0.8130453   -0.6791622
H     3.2822985    3.6802550   -0.8797716
H    -3.9849213    2.8674489   -1.1428820
H    -4.3796308    0.3183488   -0.4057986
H    -2.5716800   -4.0660404   -1.4197902
H     3.8069081   -2.0100911    0.7690203
H     2.7633297   -0.5886593   -1.7636118
H     4.0221176    1.4460571   -1.4496433
H     3.2655385    3.3318430    0.8616779
H     0.6465074    4.3939977   -0.4502444
H    -2.0283060    4.6094100    0.1582928
H    -2.0290791    4.1870050   -1.5645741
H    -3.7571685    2.9868319    0.6116638
H    -4.1467342   -2.2688675   -1.4992231
H    -4.3179232   -2.3584696    0.2616692
H    -2.6551280   -4.0911813    0.3489764
```

```
H      0.0443296   -4.4113747   -0.4998098
H      2.7987864   -4.1011739   -0.0019454
H      2.7662986   -3.5537290   -1.6918328
H      4.5246490   -2.0514737   -0.8671185
H      4.7729731    1.5218029    0.1687340
H      2.9244633    0.2751233    1.1727753
```



The reported molecular energy reported by Gaussian 09 for the DFT-B3LYP calculation was E(RB3LYP) = -2657.76765837 A.U., while for the DFT-BP86 it was E(RB-P86) = -2657.99939187 A.U.

Technical setup for using Rsockets.

System requirements.

All compute nodes must have an IP-address and hostname assigned to the Infiniband IPOverIB interface, in this case ib0. Each node thus have two IPs and DNS names.

DNS name, TCP	IP(eth0)	DNS name, IPoIB	IPoIB(ib0)
cX-Y	10.1.X.Y	cibX-Y	192.168.X.Y

Gaussian setup for Rsockets.

All that is needed to employ Rsockets as the communication layer is to insert a LD_PRELOAD environment variable in the right place of the software call chain used by the Gaussian application. The Gaussian execution environment has an environment variable named GAUSSIAN_LEXEDIR that points to the catalog where the Gaussian Linda executables reside. It turns out that this catalog only contains symlinks to the real executables residing at the same place as the non-Linda executables is installed. To get LD_PRELOAD into the right place all that is needed is to create a new catalog with a shell script and create the necessary symlinks. If GAUSS_HOME is the main installation catalog where Gaussian is installed then the following chain of commands is sufficient

```
$ cd $GAUSS_HOME
$ mkdir linda-exe-ib
$ cp /tmp/linda-rsockets.sh linda-exe-ib/
$ cd linda-exe-ib
$ for f in ../*.exel; do ln -s linda-rsockets.sh ${f##../}
```

The linda-rsockets.sh is a regular shell script executing as its calling name given by the symlink:

```
#!/bin/sh
exe=$0
ename=${exe##*/}
pname=${exe%%$ename}

export LD_PRELOAD=/usr/lib64/rsocket/librspreload.so

exec  $pname../$ename “$@”
```

After the above changes the linda-exe-ib catalog contains symlinks to the shell script which executes the right binary with the LD_PRELOAD library. The only change needed to the Gaussian environment is to change GAUSSIAN_LEXEDIR from $GAUSSIAN_HOME/linda-exe to $GAUSSIAN_HOME/linda-exe-ib.

It is also necessary to give the ib-hostnames in the input file to make Rsockets work. In the Stallo case this means replacing the hostname cX-Y with cibX-Y so the heading in the input file looks like this

```
LindaWorkers=cib1-1,cib1-2,cib1-3
NProcShared=16
```

Links

Norwegian High Performance Computing Consortium: http://www.notur.no/
High performance computing group at the University of Tromsø: http://docs.notur.no/uit
OFED Infiniband Stack: http://www.openfabrics.org
Rsockets library: http://www.openfabrics.org/downloads/rdmacm/rsockets-ofa12.pptx (Implemented by Sean Hefty, INTEL Corp.)