Halo Exchange Scalability on a Flat Switch

George W VandenBerghe

NOAA/NCEP

Oct. 31, 2006

Salient Points

- Examines scalability of finite difference exchanges and full domain transposes on a “Flat” interconnect.
- “Flat” means cpu to cpu transfer metrics are independent of other activity between other cpu pairs.
- IBM clusters are approximately “Flat” to 200 switch interfaces.. Then flatness approx begins to break down a little.
- Federation xfer rates scale linearly to 4 cpus/node, maximize at 6 or so.
- Halo exchanges are very very scalable but scalability declines with square of halo depth.
- Domain transposes are expensive, scale linearly to ~1000 processors but alltoall scaling breaks down rapidly with further taskcount increases.

Halo Exchange Scalability

- Halo exchange exchanges a grid perimeter with neighboring tasks.
- Define length **X**, memory stream rate **Mr**, Switch stream rate **Sr**, and latency **L**
- Perimeter is 4**X**
- Interior points are **X****2
- Buffer build is ~12**X /Mr** (memory stream rate) (2**X** for left and right columns, 2*5***X** for top and bottom rows which generate strided reads.
- memory sweep is **X*X/Mr**
- Xfer is 4(**X/Sr** + **L)** (latency) and exchange is 2Xfers
- latency is ~10usec. **Mr**~10**3/usec. **Sr** is **Mr**/1.3

Latency dominates for messages <10K.

HALOS

- Buffer build is ~12**X** /**Mr** (memory stream rate) (2**X** for left and right columns, 2*5***X** for top and bottom rows which generate strided reads.
- memory sweep is **X*X/Mr**
- Xfer is 4(**X**/**Sr** + **L)** (latency) and exchange is 2Xfers (one for sends and one for receives)
- Buffer extract is another 12**X** /**Mr**
- Total cost is (12X/**Mr**+4(X/**Sr** + **L**)) *2
- 24**X/Mr**+8**X**/Sr+8**L or** 8(3**X/Mr+X/Sr+L)**

Consider case where interior work scales to 1 Memcopy (cheap) or ten memcopy operations (realistic for really good code)

Number of memcopies is **Nc** and the comms:work ratio is

24**X/Mr**+8**X/Sr**+8**L : Nc*X**2/Mr**

P4 federation numbers at NCEP are

MR=1000b/usec **SR**=700b/usec **L**=10usec. Factor out **Mr**

24**X** +80/7X +8**LMr** : **X****2 (single memcopy)

24**X** +80/7**X** +8**LMr** :10**X****2 (ten memcopy)

X is ~91 for ten copy and ~300 for single copy

Simple Grid Exchange Case

- Consider 4GB domain Cell decomposition
- 5 state vbl 100 levels
- 500 8mb grids.
- If we exchange grid by grid
- **X** was 300 **X****2 was 90000 Ntasks=8K/90=~88
- **X**(10memcopy)=91 **X****2=8000
- 8M/8K=~1000

Buffer Consolidation

If we replace the grid side buffers with domain side buffers (consolidate all 500 grid sides into one buffer).

500(24X +80/7X +8LMr) : 500(X**2) (single memcopy)
500(24X +80/7X)+8LMr : 500(10X**2) (ten memcopy)

8LMr=80000 8LMr/500=160

L(1mem)=40 8e6/1600=5000
L(10mem)=6. 8e6/36=2.2e5

So by consolidating buffers this is very scalable.

Deep Halos

- For halo depth 2.   Exchange of

24X +80/7X +8LMr

Becomes

2(24X+80/7 X) +8L*Mr +8 +8L*Mr

Terms in Red represent corner point transfers.

Larger halo depth corner transfer is

8(Hd-1)**2 +8L*Mr where Hd is the halo depth.

Scalability decreases rapidly with halo depth.

Deep Halos

Roots of WX**2 +bX – c   (-b+-sqrt(b**2 +4*W*c))/2W

When C dominates X =sqrt(c/w) X decreases by 1.4 for every doubling of w.

When b dominates X ~+-b**2/2W

X ~1/W

Taskcount and scalability are inverse of X**2.

Halo depth increase H effectively divides W by H

Latency dominated case.. Scalabilty declines linearly with halo depth.

Halo interior:boundary dominated case, scalability declines with square of halo depth.

One Big Problem

- Previous analysis assumes no load imbalance.
- In NWP Physics packages this is just not true!
- Two not real good alternatives around this
- a. Dynamic decompostion, cell size depends on workload.. Tedious to code.
- b. Monte Carlo transpose.   Each cpu gets random assortement of points for the physics.
- (This presenter liked b until creating the next five slides)
- C. Tolerate it. Only true if ratio E of most expensive to least expensive points is small (<10), Toleration cuts scalabilty by this factor for latency dominated cases and SQUARE of this factor for perimeter:interior dominated cases..

Domain Transposes

- For grids, comms to work ratio was dominated by kX:X**2 term where k was fairly large (36) but overall for large enough problems, X**2 dominated.

- Domain transposes are expensive at all taskcounts but do scale (today)
- Domain transpose comms:work dominant term is kX**2:X**2. (the entire domain is moved by a transpose, not just edges) (next slides will obtain a value of 5.6 for k)

How do the Transposes run

- Domain transposes today are implemented with MPI_ALLTOALL (or alltoallv)
- Domain of size D on N processors breaks up into D/N (or d) sized pieces.
- To transpose, d is broken down into N pieces and each piece is sent to a different task. The N pieces needed for the new decomposition are then received, one from each processor.
- This method is competitive to ~10**3 tasks.

Transpose Cost

- Define DOMAIN **D**, Subdomain **sd**, taskcount **N**
- 1 Buffer build. Full sd transpose, periodic gather, cost is 5sd/Mr (5 comes from 5x slowdown doing out of cache gather)
- 2 Xfer. sd/SR or 1.3sd/Mr
- 3 Buffer extract 5sd/Mr.

- Buffer build and extract dominates cost.
- (In reality this problem does use cache better than assumed above)

Transposes

- Transpose cost 5sd +1.3sd +1.3sd +5sd.(factored out 1/Mr)

So the minimum transpose cost is 12.6 memcopy operations (?? !).

This is pessimistic and assumes we can’t somehow block the buffer build. Blocking reduces that factor of 5 to perhaps 1.5 or 2.

If we do this cost reduces to 5.6sd

Whatever that constant is it is a constant overhead at ALL taskcounts, even 2 (or 1).

This assumes no latency.

Considering Latency.

- Cost with latency is

1.5sd + N(1.3sd/N + LMr) *2 since we are dividing the subdomain sd into N pieces to send to the N other tasks.

- At high taskcount latency dominates since latency cost scales with N.
- 4GB domain example N=4000 sd=1mb sd/N=250b. 4000L=40000usec *2=80msec or .08sec Sd/Mr=.001sec. 1.3Sd/Mr=.0013.
- This problem needs very low latency or a better transpose algorithm (binary tree comes to mind)

Numbers

- Small taskcount transpose overhead is 5.6memcopy.
- Consider interior work as W mcopy ops.
- Base runtime is 5.6+W or Ba
- Define scalability limit as count where performance degrades by factor of two from base.
- NLMr=Ba*sd   = Ba*D/N
- N**2 = 1/Mr *Ba*D/L.
- For D=4GB, L=10usec, W=10, Mr=1000 Ba=15.6
- 4e9*15.6/(10*1000)=6.2e6   N=sqrt(6.2e6)=~2500
- Alternate scalability limit
- T(N) is minimum dT/dN = 0
- T=Ba*D/Mr +LN =Ba*D/(N*Mr) +LN
- dt/dn= L – Ba*D/(N**2MR) which is zero for N=sqrt(Ba*D/(Mr*L))

Alternatives??

- We don’t have the machines yet to motivate alternatives to alltoall methods.
- NCEP uses 1D decomposition in spectral space (scales to truncation/2 tasks) and threads to use ~10 cpus/task.   T384 thus scales to ~1900 cpus (190 tasks). The alltoall exchanges between 190 (not thousands) tasks.

Alternatives.

- Transpose by definition is extremely slow anyway.
- Programmers have blocked transposes and matrix multiplies since the 80s to improve cache and vector performance.
- Compilers do this too.
- This author has done it in NCEP GFS buffer build supporting Alltoall. Tripled speed of MPI domain transpose.

Something to Try

Instead of alltoall between all tasks, break tasks into blocks, do alltoalls inside the blocks, then have some master cpu in each block to alltoalls with the other masters with much larger messages.

(this is only attractive when latency dominates)

With “only” 4000 cpus on a machine and only 1000 available to single apps, this isn’t attractive yet.

Scaleability is of less interest when one runs 10**2 ensembles on 10**3 cpus (incidentally on 10**2 nodes)

Not Examined Here

- I/O
- Many to one operations.
- Single node global domain operations.
- Logarithmic or other switch topologies (this may be a big deficiency)
- Global bisection bandwidth constrained interconnects (LAN interconnect is example)

Conclusions

- Both halo exchange and domain transposes scale on current limited taskcounts (10**3)
- Halo exchanges could be scaled to 10**5 or more tasks but load imbalance in physics poses problems.
- Domain transpose scalability declines rapidly beyond 10**3 tasks.