Next Generation Subnet Manager - BGFC

OBSIDIAN STRATEGICS

March 15, 2012 - Lugano, Switzerland

Dr. David Southwell
CVO - Obsidian Strategies
Architecting for Exascale... failing with style

• System software scaling is often neglected; we tend to focus on the hardware/applications
• Moore's law - Exascale systems will likely integrate *Millions* of nodes (sorry, Carl!)

Back in the 1940s, ENIAC was built from just 17.5k vacuum tubes, and had a 50% uptime (“one five”, cf. “fives nines” !) for most of its operational life.

Exascale architectures will also push the envelope, and unlike ENIAC must be designed to tolerate node faults (soft and hard).
Architecting for Exascale... expect heterogeneity

- Compared to commercial systems, supercomputers burn twice as brightly, but for half as long
- Improvements in absolute FLOPS per (socket/ Watt/ $) drive frequent upgrade cycles
- Exascale systems are expected to be expensive (maybe, Carl!)

Traditional cluster design stresses homogeneity and uniformity...

... but it's unlikely that funding models will allow for Big Bang deployments (initial and upgrade cycles) - much more likely that there will be a need for rolling upgrade capability.

I believe we will see artificial homogeneity delivered via virtualisation.

From a fabric interconnect perspective, this is a driver for scalable support for compound topology types, and segmented InfiniBand networks are necessary to achieve this.

Obsidian Strategics Inc - HPC Advisory Council - Lugano 2012
Review InfiniBand Subnet Manager - Primary Functions

• A software component that is responsible for fabric initialisation and maintenance
• Can be hosted by a compute node, or a switch
• Most critical function is the *a priori* programming of the switch forwarding tables

Initialisation phase:

• Hardware topology discovery process (recursive queries through fabric with report back)
• Compute forwarding tables; hop-by-hop directions for every src/dst combination
• Propagate forwarding tables to all switch elements

Runtime phase:

• Track topology changes, adjust forwarding tables as required
• Respond to queries to report connectivity information at runtime
OpenSM - OFED’s InfiniBand Subnet Manager

• Commonly used Open Fabrics implementation
• Has evolved over a decade
• Multiple vendors have contributed
• Open sourced under the BSD license model

Some notable limitations:

• No support for inter-subnet routing
• Complicated internal implementation - many KLOCs, many chefs ...
• Difficult to extend (see above)
• Only “best efforts” routing engines (heuristics, no provable QOR or lack of deadlock threats)
• Cannot support compound topologies
• Just a single node active at run time - performance scaling issues
• Licensed in such a way that enhancements are not necessarily fed back into project
Introducing BGFC

- A 100% re-imagined subnet manager to address the limitations of OpenSM
- No code borrowed from OpenSM
- One architect
- Initial phase a collaboration with NASA Ames and LLNL (requirements, testing at scale)

Important characteristics:

- Mathematically perfect routing algorithm “do, or do not, there is no try”
- Cluster Implementation - N-way parallel at run-time (SA)
- Support for native InfiniBand routers
- Handles compound topologies
- Persistent database to avoid large scale re-computations
- High quality and deterministic connectivity metrics
BGFC Implementation

- Multiple SA processes (currently 7) within a node - loosely coupled, fault tolerant
- Processes have clean interfaces and independent test harnesses

Performance-sensitive runtime core is written in C++11, leveraging BGL (BOOST Graph Library)
Correctness-sensitive support code is written in Python, leveraging networkx and python-rdma*

http://www.boost.org/doc/libs/1_49_0/libs/graph/doc/index.html
http://networkx.lanl.gov/

* - python-rdma is a single-language (ok, it cheats with Pyrex) re-implementation of the entire OFED diagnostics tool set - convenient, high performance (3Gbytes/s!) , easy to integrate code.

Free, open-source, git hosted. Systematically code-gen'd (precise XML for IB structures), suitable for rapid correct development of RDMA tools in the 2.6 <= Python < 3.0 language.

If interested, please have a look at:

http://www.obsidianresearch.com/python-rdma
BGFC - the “clever” part

Optimal deadlock free InfiniBand routing is an NP problem with complexity related to the number of buffers (switches and HCAs) and no apparent short cuts ...

:-(

Subgraph Isomorphism is also an NP problem, but with complexity only related to number of switches alone, and with well known short cuts ...

So, we choose to solve the subgraph isomorphism problem and then get the routing, rather than attempt a direct assault on the routing problem!

:-)
BGFC - Algorithm sequence

1. Construct an ideal simplified graph (target topology) using Python script
2. Construct directed line graph (derivative) and create flow groups
3. Discover physical topology, treat as isomorphic subgraph of perfect target geometry
4. Perform a maximal matching between the two
5. Map the logical flow group (routing), back into physical resources
6. Execute balance weighted min hop routing on the physical flow group
7. Map the flow group topology directly onto SL2VL mapping tables
Topology Graph and Flow Groups

The Topology Graph

- InfiniBand switches become nodes in the graph
- Switch to switch connectivity represented by edges in the graph
- HCAs are ignored
- No physical port numbers
- Parallel (trunked) links are ignored

The Flow Groups (derivative)

- Directed graph capturing buffer dependencies
- Each edge in topology graph maps to two nodes in the flow group (A->B, B->A)
- Must be acyclic (and this condition will ensure resulting routing is deadlock free)
- Must be fully connected
- Boils down to SL2VL mapping tables
1-D Torus example

Topology Graph:

Flow Groups:

- d0 -> d1
- d1 -> d2
- d2 -> d3
- d3 -> d2
- d2 -> d1
- d1 -> d0
2-D Torus example

Topology Graph:

Flow Group:
Clos3 example

Topology Graph:

Flow Group:
3D hypercube flow group example
bgfc-tcmp - the topology template compiler (steps 1 and 2)

• Written in Python, using networkx for graph theoretic functions
• Python is also used as the topology description scripting language
• Generates the geometrically idealised static topology and flow groups
• Uses mathematically rigorous topology self-tests (connectivity, resiliency, acyclicnicity)

Topologies already supported by built-in templates:

Clos3  3-stage Clos network - 96, 108, 216, 288, 324 and 648-port “Clos in a box”
Clos5  5-stage Clos network (Core 3-stage Clos chassis with single chip edge switches)
FT7   7-rank Fat Tree: core and edge switches are all Clos3s
Torus  Any number of dimensions
Hypercube (isomorphic to torus case)
bgfc-tcmsg - internal QOR metrics

**Hop count metric:** The minimum number of hops it takes to communicate between any pair of access switches using the flow group graph compared with the number of hops using the topology graph. This indicates what the hop count penalty is from the deadlock avoidance scheme.

**Completeness:** The compiler attempts to add a single new legal edge to the flow group graph and checks if it is still acyclic. A flow group should contain all possible edges while still being acyclic.

**Resiliency:** The compiler will study the number of path choices that exist between each access switch pair. This metric helps identify places in the network that are vulnerable to loss of a switch link.

**Max Flow:** The compiler will study the maximum flow between all pairs in the network. This identifies the maximum available cross sectional bandwidth in the network.

**Reversibility:** The combination of SLs and LFT slices is studied to determine what sort of reversible paths could exist in the network. The IBA requires all pairs to have at least one reversible path.
bgfc-tcmp - Python scripting examples for target topology descriptions

Simple, direct from templates ...

```python
from bgfc.template import *
topology = {"my-topo": Hypercube(9)}
```

Customised from existing templates ...

```python
from bgfc.template import *
class MyNetwork(ClosTree5):
    def __init__(self):
        ClosTree5.__init__(self, core=(36, 18), cores=2, edge_conns=18//2)
    def disperse_edges(self):
        #.. insert site-specific wiring function ..

topology = {"my-topo": MyNetwork()}
```

, or arbitrarily complex by deriving from the Topology class.
bgfc-tcmp performance

All topologies: generates functional routing - **guaranteed deadlock free** and all-to-all

Hop QOR reporting shows Clos3, Clos5, FT7 are optimal

Torus/hypercube are very close to optimal (some work remaining)

624 lines of Python, compared to ~17,200 lines in OpenSM!!

Mathematically provable correctness, not “faith-based” correctness
bgfc-tcmp performance

All topologies: generates functional routing - **guaranteed deadlock free** and all-to-all

Hop QOR reporting shows Clos3, Clos5, FT7 are optimal

Torus/hypercube are very close to optimal (some work remaining)

624 lines of Python, compared to ~17,200 lines in OpenSM!!

Mathematically provable correctness, not “faith-based” correctness
BGFC persistent database mechanism

Always with a view to Exascale systems, at all costs we need to avoid unnecessary large-scale recalculations that the BGFC architecture allows you to make beforehand - the topology databases are cached to persistent storage and distributed:

- Consistent “network orientation” characteristics
- Fast fabric initialisation (verbs + mcast accelerated scan to confirm topology → pre-load)
- Incremental topology changes can usually be accommodated without system-wide re-calc
- Ability to accurately reproduce prior network configurations
Inter-subnet routing

As providers of wide area InfiniBand devices, Obsidian saw the need for routing between multiple subnets first, and first generation hardware has supported it since 2006.

Subnet manager support for routing is a major motivation for BGFC.

**Longbow E100** - global reach, AES encrypted, routed 4X SDR IB over DWDM or 10GbE WANs

Chapter 19 of the InfiniBand specification stopped before describing inter-subnet manager communication, and no existing subnet manager supports it.

BGFC will fill this need, and Obsidian is working with IBTA in this endeavour.
Drivers for multiple subnet fabrics - politics, performance, availability
Drivers for multiple subnet fabrics – politics, performance, availability
Drivers for multiple subnet fabrics - politics, performance, availability
Drivers for multiple subnet fabrics - politics, performance, availability

Subnet 1 - 4D Hypercube (Pleiades elements)

Subnet 2 - Clos3 fabric (Hyperion elements)

Switches in Subnet 1

Switches in Subnet 2

Encrypting range-extending routers (Obsidian Longbow E100s)
BGFC and multiple subnets

• Simply treat routers as switches for topology analysis
• Aggregate all subnets into a single giant topology template
• Globally compute the flow groups
• Re-partition back into separate subnets

Subnets can then operate independently - for as long as their topologies remain subgraph isomorphic to the giant template.

Optimal routing solution - full bandwidth, 100% deadlock potential free.
SA (Subnet Administrator) cluster Implementation

- Performance scalability
- High availability

Implementation strategy:

- Built correctness first using multiple processes in a single node
- Using a “write-once” shared memory IPC system
- Extended to multiple node implementation; architecture leads to lockless design
- Designed for node (not switch) hosting, so `verbs+mcast` can be used to accelerate SA functions
- All BGFC nodes concurrently active, but only one node is **WODB** writer at any time per subnet
BGFC - Obsidian's motivations

- BGFC is licensed under GNU GPL V2
- Designed to be hosted by server nodes, not embedded in proprietary switches
- Committed to standards based approach
- Facilitates adoption of other Obsidian products...
Upcoming Obsidian hardware - 6-port 40G InfiniBand router

Crossbow R400-6:
Six QDR 4X IB ports
Ethernet/ RS-232 OOB management ports
Enhanced link-level buffering for extended optical links

• ¼ 1RU form-factor
• <400ns latency injection
• 48 Gbytes/s between subnets per 1RU shelf
Upcoming Obsidian hardware - 40G light path range extender product

**Longbow C400:**
- QSFP 4X SDR/DDR/QDR InfiniBand port
- 1, 2 or 4 SFP+ 10Gλ optical modules
- Ethernet/ RS-232 OOB management ports
- Up to 80km range - well suited to WDM WANs (skew considerations)
- Integrated FEC on optical paths (fits within 8B/10B overhead space)
- Graceful QDR → DDR → SDR LAN degradation if WAN λs fail

• ¼ 1RU form-factor
• <600ns latency (+ 5.02µs / km @ c v1.0)
• 16 Gbytes/s per 1RU shelf
WAN support device for Longbows - Coarse Wavelength Division Multiplex

Accessory A-CWDM81:  
Eight Channelised LC duplex ports +1310nm ↔ Combined LC duplex port 1310nm + 1470-1610nm range λs  
Standard ITU 100GHz frequency grid spacing (20nm)  
< 2dB optical insertion loss (worst case)

- ¼ 1RU form-factor
- <3ns latency injection
- 36 Gbytes/s per 1RU shelf onto 4 SMF pairs
- Passive device - 0W
Wrap Up

BGFC is being developed as a new subnet manager architecture, to support large-scale, distributed and/or high criticality applications with a number of new features:

- GPLv2 license model
- Novel application of graph theory to significantly improve:
  - Routing calculation speed
  - Confidence in routing quality (to 100% - mathematical rigour)
- Parallel N-way active implementation (performance, availability)
- Persistent topology databases
- Easy to use Python scripting model for topology description
- Support for native InfiniBand inter-subnet routing hardware

Anticipated initial public release in 2012 - we are currently working with (as yet unnamed) additional testing collaborators.
Thank you!