Porting HPC Tools to FreeBSD

Brooks Davis
The Aerospace Corporation
El Segundo, CA
brooks@aero.org

Abstract

Since 2001 we have used FreeBSD as a high performance computing (HPC) cluster operating system. In the process we have ported a number of HPC tools including Ganglia, Globus, Open MPI, and Sun Grid Engine. In this paper we discuss the process of porting these types of applications and issues encountered while maintaining these tools. In addition to general issues of porting code from one Unix-like operating system to another, there are several types of porting common to many HPC tools which we will explore. Beyond porting, we will discuss how the ports collection aids our use of HPC applications and ways we think overall integration could be improved.

1 Introduction

At The Aerospace Corporation we have designed, built, and operated a FreeBSD based HPC cluster since 2001[Davis]. Figure 1 shows the Fellowship cluster in it's current form with 352 dual processor nodes. In the process of building and running Fellowship we have ported a number of open source HPC tools to FreeBSD. In this paper we discuss the process of porting them, issues we encountered, and a few pet peeves about application portability.

We begin with an overview of the porting of three tools: Sun Grid Engine—also known as SGE—a batch job manager; the Ganglia monitoring system, a cluster/grid monitoring tool; and Open MPI, a leading implementation of the Message Passing Interface which is an API for message based parallel programming. After discussing these tools we review a number of general portability issues and possible remedies for FreeBSD.

We hope to demonstrate that porting HPC tools is straight-forward and encourage others to join the effort to port more tools. We also want to encourage developers to think of ways to make FreeBSD an easier porting target.

Figure 1: Fellowship Circa February 2007

2 Experiences Porting Applications

2.1 Ganglia

The Ganglia monitoring system[Massie]—usually referred to as Ganglia—is a cluster and grid monitoring tool that provides current and historical data on the status of a cluster or collection of systems. The data includes CPU count, CPU utilization, memory use, and network I/O among others. Ganglia is used on systems ranging from small clusters to PlanetLab[PlanetLab], a globally distributed network of over 1000 nodes at nearly 500 sites. Beyond HPC applications Ganglia is used to monitor web farms or even desktop computing resources.

Ganglia's architecture consists of monitoring daemons (gmond) on each node that send periodic updates of various metrics to a central collector daemon (gmetad) that stores the results in a set of Round Robin Databases[RRDtool]. The metrics cover many aspects

of system configuration and load, and may either be fixed (per-boot) or vary with time.

Core metric providers are implemenated as functions in the monitoring daemon that return string or numeric values. In early versions of Ganglia the set of metrics was fixed at compile time. In more recent versions, it is configured at runtime and pluggable metric providers are supported including Python modules.

The process we used to port metrics is simple. First, we checked the code for other operating systems to determine what the metric means. Then we figured out a way to use standard tools such as top, ps, procstat, or sysctl to extract that information from the system. Once appropriate tools were identified, we examined their source code to determine how they retrieved the information in question and used that information along with man pages to write an appropriate metric implementation. In many cases it was possible to copy the code directly from FreeBSD utilities along with another copyright statement and license block because Ganglia is BSD licensed.

One of the simplest metrics to port was cpu_num which is the number of CPU cores in a machine. Listing 1 shows the entire implementation and illustrates the basics of Ganglia metric implementation. Each metric is implemented as a function that returns a g_val_t structure representing the current value of the metric.

Listing 1: cpu_num_func()

```
g_val_t
cpu_num_func(void)
{
  g_val_t val;
  int ncpu;
  size_t len = sizeof (int);
  if (sysctlbyname("hw.ncpu", &ncpu,
   &len, NULL, 0) == -1 || !len)
    ncpu = 1;

  val.uint16 = ncpu;
  return val;
}
```

The swap_total and swap_free metrics are more complex metrics and cover several interesting cases including the use of the kvm[FreeBSD-kvm] interface and supporting multiple interfaces for cross platform support. In porting them we observed that users and administrators can obtain swap information from the swapinfo(1) command[FreeBSD-pstat]. We dug into the code and determined that it used kvm to access the data if called on a coredump and a sysctl if not. However, when we first ported Ganglia, FreeBSD 4 did not support the sysctl interface so we supported both sysctl and kvm methods because the sysctl interface allowed us to run Ganglia without ever being root, but the kvm interface was needed since we need to support FreeBSD 4.x. As in Listing 2 our implementation probes at runtime and prefers the sysctl interface where available.

Listing 2: Excerpts from metric_init()

```
g_val_t
metric_init(void)
{
  g_val_t val;

  mibswap_size = MIB_SWAPINFO_SIZE;
  if (sysctlnametomib("vm.swap_info",
   mibswap, &mibswap_size) == -1) {
    kd = kvm_open(NULL, NULL,
     NULL, O_RDONLY,
     "metric_init()");
  } else {
    kd = kvm_open(_PATH_DEVNULL,
     NULL, NULL, O_RDONLY,
     "metric_init()");
    use_vm_swap_info = 1;
  }
  pagesize = getpagesize();
  <...>
  val.int32 = SYNAPSE_SUCCESS;
  return val;
}
```

The core of the swap_total implementation shown in Listing 3 uses the result of the probe in metric_init() to determine how to retrieve the amount of swap used using sysctl or kvm. In the sysctl case, we retrieve information from each individual swap device and total them where the kvm interface provides a direct total. There are a couple other interesting things of note here. First, Ganglia expects swap size to be a floating point number of KiB. Variants of this are common in Ganglia due to the desire to represent large sizes identically on all machines. Second, is the XSWDEV_VERSION line. This points out a problem with version numbers in binary interfaces. They are good in principle, but clients need to know about the new version to do anything with it so in practice they do little to help provide ABI stability. In this case we have no choice but to reject versions we do not understand so existing binaries would necessarily be broken if this were changed.

Listing 3: Excerpts from swap_total_func()

```
if (use_vm_swap_info) {
  for (n = 0; ; ++n) {
    mibswap[mibswap_size] = n;
    size = sizeof(xsw);
    if (sysctl(mibswap,
     mibswap_size + 1, &xsw,
     &size, NULL, 0) == -1)
```

```
        break;
      if (xsw.xsw_version != XSWDEV_VERSION)
      return val;
      totswap += xsw.xsw_nblks;
    }
} else if (kd != NULL) {
    n = kvm_getswapinfo(kd, swap, 1, 0);
    if (n < 0 || swap[0].ksw_total == 0)
      val.f = 0;
    totswap = swap[0].ksw_total;
}
val.f = totswap * (pagesize / 1024);
```

The Ganglia metrics for memory use were some of the hardest to port and some of the ones we are least satisfied with overall. The problem stems from the fact that the different virtual memory systems in different operating systems use vastly different sets of accounting buckets. The Ganglia metrics "total", "buffers", "cached", "free", and "shared" appear to be derived from values that are easy to obtain on Linux. Unfortunately, they do not match any of the memory use statistics provided by FreeBSD. For example top shows "active", "inactive", "wired", "cache", "buffers", and "free" memory, where "free" memory is always a small number on a system that has been up and active for some time since the VM system sees little need to waste CPU time freeing memory that might be used again. As the saying goes, free memory is wasted memory in the operating system. Even total memory is complicated. For example we have both `hw.physmem` which is the actual amount of memory in the system (modulo 32-bit limits on 32-bit, non-PAE system) and `hw.realmem`, the memory that it is possible to use. Table 1 shows the set of mappings we chose, but this is far from optimal.

Over all Ganglia was a straight forward package to port. Supporting multiple FreeBSD versions required some effort, but most metrics were easy to support.

2.2 Sun Grid Engine

Sun Grid Engine (SGE)[SGE] is a leading open source batch scheduler and resource manager. The other credible option is Torque, a fork of OpenPBS. Prior to Sun's purchase of Grid Engine and the subsequent open source release, we had been attempting to get OpenPBS to work in our environment, but it was never stable for us (or many others based on traffic on the on the OpenPBS mailing lists). When SGE was released as open source in 2001 and Ron Chen started a port we leap at the chance to try something else and completed the port.

Ganglia Metric	*sysctl Used in Port*
mem_total	hw.physmem
mem_buffers	vfs.bufspace
mem_cached	vm.stats.vm.v_cache_count
mem_free	vm.stats.vm.v_free_count
mem_shared	0

Table 1: Ganglia Metric to sysctl Mappings

The first step in porting SGE was figuring out the build system. It is a unique and complex system consisting of a nearly three-thousand line csh script named aimk (not be confused with the aimk build tool from PVM[PVM]). Within the script, each pair of operating system and architecture has a separate configuration section. The script invokes a dependency generator, a variety of make instances, `configure` in some cases, and in recent versions Apache Ant. Historically `configure` was prerun for each platform and thus `configure` was not actually run by aimk.

In the process of porting the build system we introduced a number of innovations to reduce the complexity of adding new platforms. Most of these were related to adding support for multi-platform builds since we knew from the start we wanted to support at least FreeBSD i386 and amd64. We wanted to be able to support new architectures with little or no additional change to the build system. To that end we added a single configuration section for FreeBSD architectures and changed the architecture naming so that FreeBSDs platform string is always fbsd-*arch* where *arch* is the machine architecture as given by `uname -m`. We also augmented the build system so that for portions of the source tree that depend on configure output, we run configure if no pregenerated output is available.

Because the build system requires many steps to generate a working system, we also created a FreeBSD port[Ports] early on so we can build the system repeatably with a reasonable number of command invocations. This greatly simplified the process of testing new features.

One section we have not yet ported is the code to build the Java interfaces. Java use started with an interface to the DRMAA job submission interface, but now includes a GUI installer. At one point we were able to harvest Java components from the pre-build binaries, but those are no longer available under an appropriate license so we have disabled all Java support in the port.

After the build system, we tackled a variety of portability definitions. Early versions of Grid Engine were ported to systems that significantly predate modern versions of the C standard such as C99. As a result there are several sets of type definitions for fixed width

integers as well as a number of definitions to handle differences in handling required to print a number of standard Unix typedefs like uid_t, gid_t, and pid_t. For example these are some of the FreeBSD specific definitions are shown in Listing 4.

Listing 4: cpu_num_func()

```
#define u_long32   uint32_t
#define uid_t_fmt  "%u"
#define gid_t_fmt  "%u"
#define pid_t_fmt  "%d"
```

A fair bit of this code could be simplified today, but doing so would probably require removing support for some older systems like Cray and HP-UX and the developers have not been willing to do so.

Like Ganglia, SGE collects a number of metrics from the execution daemons on each node and uses those results to make job placement decisions. The implementation is not as neatly divided into metric functions, but the basic principles of porting metrics are the same so we will not cover them here. We found the easiest way to find them was to search for LINUX in the source tree to find all the things Linux had to implement. The Darwin and NetBSD ports seem to have searched for FREEBSD to aid their porting.

One place where SGEs metrics differ from Ganglia is that SGE wants to track resource use by all the processes that make up a job. This is done in the Portable Data Collector sub system. On Irix, there is a mechanism for attaching job IDs to processes and then querying resource use for the process and all its children (as well as sending signals to the group). This feature does not exist on most other platforms so SGE implements a clever hack to achieve a similar effect. The trick is that they allocate an otherwise unused group on each node and then add it to the group list for each process. Since the group list is copied to each child on fork() and ordinary users can not adjust their group list this provides a tag which can be used to detect the processes that make up a job. When SGE wants to determine the cumulative resource use of a job, it walks the process table using kvm and totals all resource use. Recent changes to FreeBSD (available in 7.3 and 8.0) will allow the use of sysctl to perform this task, but that will not be a portable option for some time.

The SGE port took some significant work and still could use some more polish, but the individual pieces were relatively simple to port and the current result is quite useful.

2.3 Open MPI

Open MPI[Open MPI] is a leading open source implementation of the Message Passing Interface[MPI]. MPI is the primary interface for building message passing parallel application. These are applications where multiple process coordinate their computations with messages. MPI hides the details of the underlying network beneath a common API. Open MPI is also the basis of commercial toolkits such as Sun's HPC toolkit.

Overal, Open MPI was one of the easiest ports we have made. The code is highly modular with extensive use of autoconf scripts to detect features. It is clear the development team has taken to heart the historical need for portability in HPC code, especially middleware. The one feature we needed in the initial port was the backtrac() and backtrac_symbols() functions glibc implements. Fortunately the devel/libexecinfo port already provides that functionality so we simply added the necessary autoconf bits to do detect it and we had a working port.

One optional feature we have not yet ported is CPU affinity. The Open MPI team created the Portable Linux Processor Affinity (PLPA) framework to deal with ABI issues in the Linux CPU affinity system calls. We could easily support the same interface without the tricks they play in the name of binary compatibility, but have not done so yet. PLPA has been superseded in Open MPI by a combination of its functionality and libtopology[libtopology] in a new package called Portable Hardware Locality (hwloc), but porting PLPA may still be useful because it has been incorporated into recent SGE releases.

3 Porting Issues

3.1 IPv6 Socket Behavior

One of the more annoying porting issues we encountered in Ganglia was with IPv6 support. KAME derived IPv6 stacks violate RFC 2553 and disallow IPv4 mapped IPv6 sockets from receiving IPv4 data and instead require two sockets to be opened. This is done for relatively well justified security reasons[Metz], but can be a significant portability issue. When Ganglia introduced IPv6 support they did so using the Apache Portable Runtime[APR] which did not understand this issue. As a result, IPv6 support was disabled on FreeBSD for some time. After many discussions, Ganglia was modified to open two sockets as

required under FreeBSD.

Given that code tends to expect this behavior to work and that changes to the RFCs have not been forthcoming, we think the BSD default should be reevaluated.

3.2 Signal Handling

Another interesting portability issue we have encountered was with signal handlers. At some point the SGE developers fixed some bugs by adding persistent signal handlers. Unfortunately, they used the easy to use, but not entirely portable sigset() function. POSIX defines the sigset() function, but FreeBSD does not yet implement it. In practice our implementation of signal() is equivalent to sigset(), but POSIX compliant implementations are not. At the time we were able to persuade them to switch all instances of sigset() or signal() to sigaction() which is well defined, but has the unfortunate characteristic of requiring multiple lines of code to replace each simple signal() or sigset() call. Since then further sigset() calls have been introduced.

The overall state of affairs with respect to signal handling is a mess. This quote from Jim Frost[Frost] sums up the situation:

> To make certain that no one could write an easily portable application, the POSIX committee added yet another signal handling environment which is supposed to be a superset of BSD and both System-V environments.

The POSIX sigaction() function is well defined, but too hard to use. FreeBSD needs to follow NetBSD and implement sigset() and related functions.

3.3 Schema Conflicts

When porting Ganglia one of the more vexing issues we ran into was matching the Linux derived schema to represent memory use with the available metrics in FreeBSD. This is a common problem in monitoring systems, some other examples include: Network protocol statistics where different implementations may present much more or less information or present the same information is different ways. Determining the disk space used or available on disks and wanting to distinguish between local, remote, and pseudo file systems. Programs tend to end up with hard coded lists of local file system types. This scales poorly and fails to answer questions like, "Are UFS file systems on a SAN local or remote?"

As porters there often is little we can do about an existing schema. The best we can do is encourage standard and rational schemas where possible. A good source of well though out schemas is SNMP MIBs[RFC3411].

3.4 Lack of Good Interfaces

One final general portability issue is a lack of good interfaces to some operating system data. For example an SGE port would have been much simpler if there were a way to track the resource of a process and all its children and to send signals to all of them such as the Irix jid_t. An interface that does not require access to kernel memory is now available, but is not particularly efficient. Another program that suffers from these problems is lsof[Abell] which currently accesses kernel memory directly to retrieve all manner of file handle related data. As a result it breaks on a regular basis when kernel developers move things around. Creating proper interfaces to these sorts of data would be useful, but it is often difficult to determine which needs to be exposed when developing features. Additionally, poorly thought out interfaces can lead to maintenance problems down the road as interfaces like the routing socket demonstrate.

In FreeBSD is adding new interfaces at a steady rate and is developing a more mature understand of interface design. Some areas still need work and a number of interfaces need significant modernization, but the project is heading in the right direction.

4 Conclusions and Future Work

We have covered our ports of the high performance computing tools Ganglia, Sun Grid Engine, and Open MPI to FreeBSD. In all three cases the basic task of porting was relatively straight forward. Where we encountered problems we noted how we resolved major issues and have discussed ways FreeBSD could be changed to ease future porting efforts.

We believe that porting HPC tools to FreeBSD is something more people could do and we encourage others to jump in and help out. Some tools that would be beneficial to port include: PLPA and its successor hwloc; various PAPI[PAPI] based performance analysis tools; the Eclipse Parallel Tools Platform[PTP]; and the ROCKS cluster system[Rocks]. The porting of PAPI based tools could provide a significant competitive advantage to FreeBSD as the mainline Linux kernel and most Linux distributions to not include na-

tive support for hardware performance counters, but FreeBSD ships with hwpmc[FreeBSD-hwpmc] enabled by default.

References

[Abell] Vic Abell, *Frequently Asked Questions about lsof*. ftp://lsof.itap.purdue.edu/pub/tools/unix/lsof/FAQ. January 18, 2010.

[APR] The Apache Portable Runtime Project. http://apr.apache.org/

[Davis] Brooks Davis, Michael AuYeung, J. Matt Clark, Craig Lee, James Palko, Mark Thomas, *Reflections on Building a High-performance Computing Cluster Using FreeBSD*. Proceedings, AsiaBSDCon 2007.

[FreeBSD-hwpmc] The FreeBSD Project, hwpmc(4) *FreeBSD Kernel Interfaces Manual*, http://www.freebsd.org/cgi/man.cgi?query=kvm&manpath=FreeBSD+8-current September 22, 2008.

[FreeBSD-kvm] The FreeBSD Project, kvm(3), *FreeBSD Library Functions Manual*, http://www.freebsd.org/cgi/man.cgi?query=kvm&manpath=FreeBSD+8-current January 29, 2004.

[FreeBSD-pstat] The FreeBSD Project, pstat(8), *FreeBSD System Manager's Manual*, http://www.freebsd.org/cgi/man.cgi?query=pstat&manpath=FreeBSD+8-current August 20, 2008.

[Frost] Jim Frost *UNIX Signals and Process Groups*. http://www.cs.ucsb.edu/~almeroth/classes/W99.276/assignment1/signals.html August 17, 1994.

[libtopology] libtopology http://libtopology.ozlabs.org/

[Massie] Matthew L. Massie and Brent N. Chun and David E. Culler, *The Ganglia Distributed Monitoring System: Design, Implementation And Experience*. Parallel Computing. Volume 30. 2003.

[Metz] Craig Metz and Jun-ichiro itojun Hagino, *IPv4-Mapped Addresses on the Wire Considered Harmful*. http://tools.ietf.org/id/draft-itojun-v6ops-v4mapped-harmful-02.txt October 21, 2003.

[MPI] Message Passing Inteface Forum http://www.mpi-forum.org/

[Open MPI] Open MPI http://www.open-mpi.org/

[PAPI] PAPI: Performance Application Programming Interface. http://icl.cs.utk.edu/papi/

[PlanetLab] PlanetLab: An open platform for developing, deploying, and accessing planetary-scale services. http://www.planet-lab.org/

[Ports] The FreeBSD Collection Ports http://www.freebsd.org/ports/index.html

[PTP] PTP - Parallel Tools Platform. http://www.eclipse.org/ptp/

[PVM] PVM: Parallel Virtual Machine http://www.csm.ornl.gov/pvm/

[RFC3411] D. Harrington, R. Presuhn, B. Wijen, *An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks*. RFC 3411. Internet Engineering Taskforce, Network Working Group. December 2002.

[Rocks] Rocks Clusters. http://www.rocksclusters.org

[RRDtool] RRDtool http://oss.oetiker.ch/rrdtool/

[SGE] Sun Grid Engine Project http://gridengine.sunsource.net/

[FreeBSD-cpuaff] The FreeBSD Project, cpuset_getaffinity(2), *FreeBSD System Calls Manual*, http://www.freebsd.org/cgi/man.cgi?query=cpuset_getaffinity&manpath=FreeBSD+8-current March 29, 2008.

[FreeBSD-cpuset] The FreeBSD Project, cpuset(2), *FreeBSD System Calls Manual*, http://www.freebsd.org/cgi/man.cgi?query=cpuset&manpath=FreeBSD+8-current March 29, 2008.