LA-UR-21-26591
Approved for public release; distribution is unlimited.

Title: Compiling MCNP6.2 for ARM Clusters

Author(s): Grieve, Tristan Sumner

Intended for: 2021 MCNP User Symposium, 2021-07-12 (Los Alamos, New Mexico, United States)

Issued: 2021-08-11 (rev.1)

Los Alamos NATIONAL LABORATORY

Compiling MCNP6.2 for ARM Clusters

Avery Grieve

XCP-3 (Monte Carlo Codes)

2021 MCNP User Symposium
July 13, 2021

Topics

1. Motivation
   - Why bother with this?
2. Preliminary Cluster Setup
   - What do you need to set up a cluster from scratch?
3. Compilers and Options
   - gfortran, gcc, g++, OpenMPI
4. Modifications to configuration file
   - x86-64 vs aarch64 compiler options
5. Performance
   - Comparing OMP and MPI runs against an x86-64 system
6. Drawbacks & Considerations
   - Nothing is perfect

Motivation

- ARM processors are increasingly common:
  - Supercomputer Fugaku: A64FX 48C (7.63 M cores)
  - New Apple silicon
  - Raspberry Pi and other off-the-shelf single board computers (SBCs)

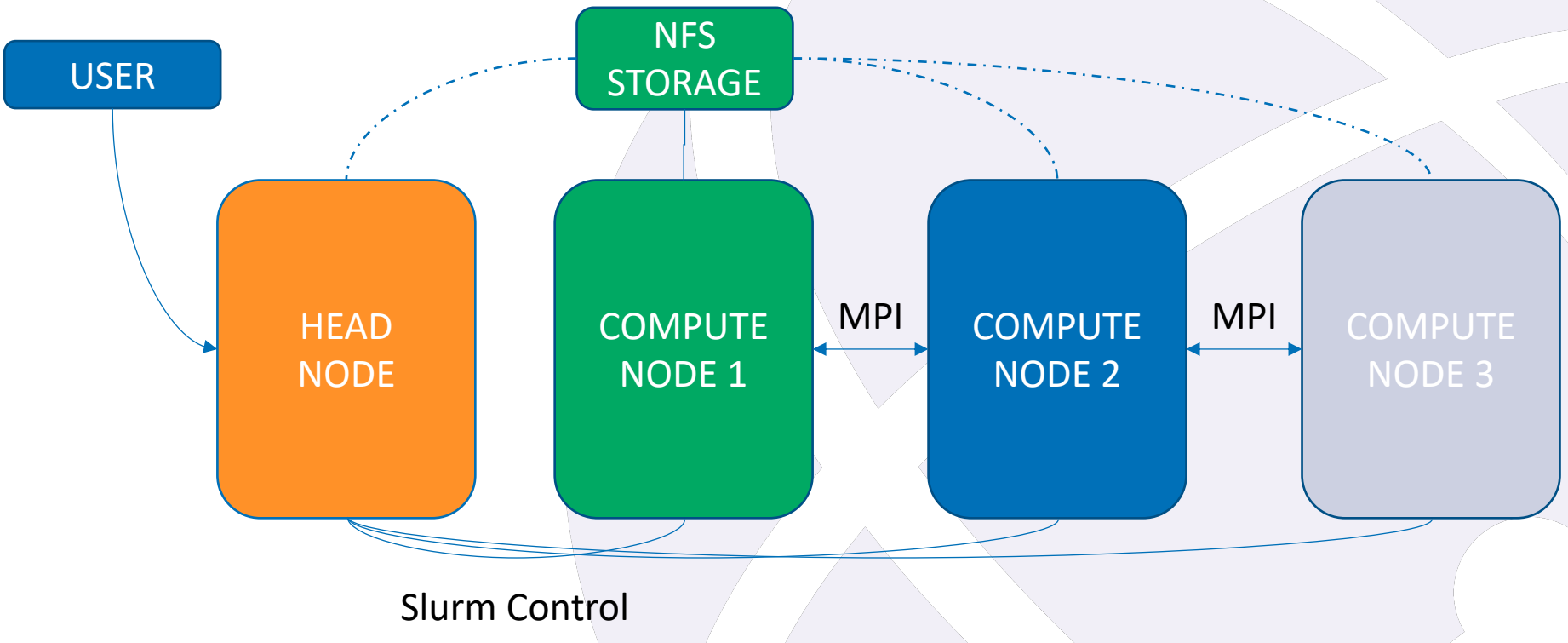
- Single board computers are low cost devices with strong potential in research applications, whether as a local testbed for developing parallel applications or a low-cost computational cluster

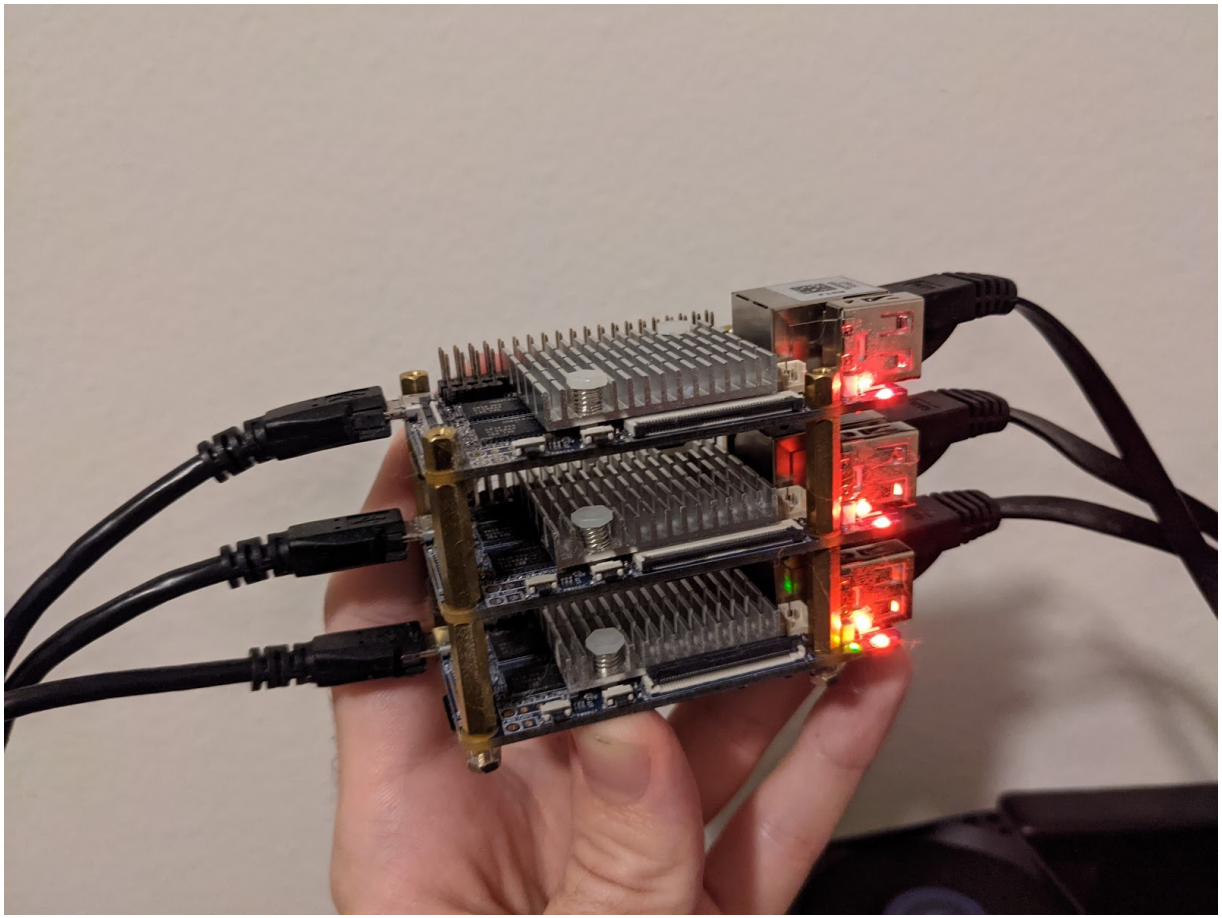
- Potential for recycling old smartphones utilizing ARM ISAs into cluster-like applications (reduction in e-waste).

Preliminary Cluster Setup

- When building a cluster from scratch, there are several key considerations:
  - Lightweight operating system
    - Disable functions that aren't needed, consider headless setup to avoid display overhead, etc
  - Interconnect type
    - Ethernet interconnect via a switch is probably the easiest (only need to configure /etc/hosts), faster options may be compiled
  - Networked storage between cluster devices for MCNP installation
    - nfs-common package (Debian)
  - Highly recommended to install Munge & Slurm for job scheduling & user management
    - Installation and setup is beyond the scope of this presentation. Many distributions have both a Slurm and Munge package.
    - Recommend setting up a head node independent of the compute nodes.

End Goal


USER
NFS STORAGE
HEAD NODE
COMPUTE NODE 1
MPI
COMPUTE NODE 2
MPI
COMPUTE NODE 3
Slurm Control



Personal testbed cluster with an OrangePi PC2 head node (not pictured) and three NanoPi Fire 3 compute nodes (24 threads @ 1.4GHz total). Ethernet interconnect. Low power usage (<50W max). This cluster is the basis of this original work.

Compilers & Requirements for building with CONFIG='mpi omp gfortran'

- LA-UR-17-30373 is a good place to start for compiling. However, versions of gfortran newer than 7.1 do build the code.
  - Check the operating system's package repositories for a version of gfortran that will build the code.
  - The latest version of gfortran tested during this work is 9.3.0

- Ensure the "build-essential" package or equivalent is installed for gcc and g++.

- The version of OpenMPI shouldn't matter, but each node needs the same version compiled. Recommend compiling from scratch and installing to default location.
  - Latest tested for this work is OpenMPI 4.0.5

Modifying the configuration file

- Extract the DVD folders to shared directory & navigate to Source directory:
  - $ cd /cloud/MCNP/MCNP_CODE/MCNP620/Source
- Trying to build MCNP now will lead to invalid compiler option errors, so the Linux config file must be modified
- Open Linux.gcf file in the Source/config/ directory:
  - $ vim config/Linux.gcf

```
┌[agrieve@MBP]-(MCNP620/Source)-
└> ls config
AIX.gcf         Darwin.gcf      Linux.gcf       SunOS.gcf       Windows.gcf     make_setup.mk   version_info.pl
AIX_aux.gcf     IRIX64.gcf      OSF1.gcf        VC_info.gcf     make_depends.pl mcnp_env.pl
┌[agrieve@MBP]-(MCNP620/Source)-
└> vim config/Linux.gcf
```

Modifying the configuration file

- Locate gfortran compiler section in configuration file (line 253)

```
# ---------------------------------------------- GFORTRAN
ifeq (GFORTRAN,$(filter GFORTRAN,$(FCOMPILER)))
# gmake has a default of f77 set for FC .... Check to see if it's still default
# if its set to default and gfortran is requested then try to find gfortran in
# the users $PATH
  FC = gfortran

  FCPU            ?= -m64
  CCPU            ?= -m64
  ifeq (memmodel,$(filter memmodel,$(CONFIG)))
     FMEMMODEL       ?= -mcmodel=medium
     CMEMMODEL       ?= -mcmodel=medium
  endif
  FC_FLAGS        ?= -mieee-fp -fdollar-ok
```

- Problematic flags are circled… What are these and what are the ARM gfortran equivalents?

Modifying the configuration file

X86 Compiler Option	Description of Option	AARCH64 "Equivalent"	Description of Equivalent
-m64	"The -m64 option sets int to 32 bits and long and pointer types to 64 bits, and generates code for the x86-64 architecture" [1]	-mabi='lp64'	"Generate code for the specified data model. Permissible values are…and 'lp64' for SysV-like data model where int is 32 bits, but long int and pointers are 64 bits." [2]
-mieee-fp	"Control whether or not the compiler uses IEEE floating-point comparisons. These correctly handle the case where the result of a comparison is unordered. " [1]	-march='native' (activates fp feature modifier)	"The value 'native' is available on native AArch64 GNU/Linux and causes the compiler to pick the architecture of the host system." "Enable floating-point instructions. This is on by default for all possible values for options -march and -mcpu." [2]

[1] https://gcc.gnu.org/onlinedocs/gcc/x86-Options.html

[2] https://gcc.gnu.org/onlinedocs/gcc/AArch64-Options.html

Final Linux.gcf

- Replace x86-64 options with new aarch64 options:

```
ifeq (GFORTRAN,$(filter GFORTRAN,$(FCOMPILER)))
# gmake has a default of f77 set for FC .... Check to see if it's still default
# if its set to default and gfortran is requested then try to find gfortran in
# the users $PATH
  FC = gfortran

#  FCPU            ?=-m64
#  CCPU            ?=-m64
  FCPU ?= -mabi='lp64'
  CCPU ?= -mabi='lp64'
  ifeq (memmodel,$(filter memmodel,$(CONFIG)))
     FMEMMODEL       ?= -mcmodel=medium
     CMEMMODEL       ?= -mcmodel=medium
  endif
  FC_FLAGS       ?= -fdollar-ok -march='native'
```

Build, Set Environment Variables, Test

In the MCNP620/Source directory, run:

- `make realclean`
- `make build GNUJ=N CONFIG='omp mpi gfortran' *`
- `export DATAPATH=/path/to/MCNP_DATA`
- `export OMP_STACKSIZE=128M (IMPORTANT)`
- `export PATH=/path/to/MCNP620/bin/:$PATH`
- `make test CONFIG=mpi`

Note: Building an OMP-only executable after building an mcnp6.mpi executable will overwrite the MPI version but building with MPI after building an OMP version will not overwrite the OMP version giving two executables in `bin`: mcnp6 & mcnp6.mpi

* See LA-UR-17-30373 for a list of CONFIG options

Regression test results:

CASE	OUTP diff	MCTAL diff	WWOUT diff	PLOT/ PTRAC diff	TEXT1/ MESHTAL diff	TEXT2/ EEOUT diff	TEXT3/ GMV diff
inp23	388	–	–	–	–	–	–
inp87	222	–	–	–	–	–	–
inp93	27944	1868	–	–	–	–	–
inp94	386	–	–	–	–	–	–
inp131	–	–	–	–	8498	–	–
inp84	49423	9709	–	–	–	–	–
inp96	–	–	–	–	1404	–	–
inp110	49284	9710	–	–	–	–	–
inp134	8778	5196	–	–	–	–	–
inp135	2340	158	–	–	–	–	–
inp1004	22920	1032	–	–	–	4648	–
inp1008	–	–	–	–	–	55	–
inp1009	–	–	–	–	–	55	–
inp1011	872	–	–	–	–	–	–
inp1015	20756	194	–	–	–	4510	–
inp1018	47198	256	–	–	–	52661	–
inp1030	–	–	–	–	–	292	–
inp1031	2496	–	–	–	–	3188	–
inp1034	3620	–	–	–	–	442	–
inp1035	12363	352	–	–	–	1072520	–
inp1036	354	–	–	–	–	–	–
inp1040	30774	256	–	–	–	13508	–
inp1042	23835	852	–	–	–	41397	–
inp1055	1446	–	–	–	–	–	–
inp1063	53586	312	–	–	–	33243	–

```
1482c1482
<   9.035  E-09 0.0     9.056  E-09 0.0     9.077  E-09 0.0     9.097  E-09 0.0
---
>   9.035  E-09 0.0     9.056  E-09 0.0     9.076  E-09 0.0     9.097  E-09 0.0
2368c2368
<   5.283  E-09 0.0     5.295  E-09 0.0     5.307  E-09 0.0     5.319  E-09 0.0
---
>   5.283  E-09 0.0     5.295  E-09 0.0     5.307  E-09 0.0     5.320  E-09 0.0
2437c2437
<   9.974  E-09 0.0     9.998  E-09 0.0     1.002  E-08 0.0     1.004  E-08 0.0
---
>   9.974  E-09 0.0     9.997  E-09 0.0     1.002  E-08 0.0     1.004  E-08 0.0
```

Of 1063 regression tests, 25 have reported differences (~2.3%)

Differences in MCTAL files are most important. These appear small, but are worth noting.

```
<   1.047  E-06 0.13     5.727  E-07 0.13     8.131  E-07 0.13     1.148  E-06 0.13
<   1.685  E-06 0.13     2.419  E-06 0.13     3.494  E-06 0.13     5.100  E-06 0.13
<   7.554  E-06 0.13     1.108  E-05 0.13     1.629  E-05 0.13     2.388  E-05 0.13
<   3.518  E-05 0.13     5.156  E-05 0.13     7.576  E-05 0.13     1.076  E-04 0.13
<   1.549  E-04 0.13     2.193  E-04 0.13     2.987  E-04 0.12     3.805  E-04 0.13
<   4.579  E-04 0.14     4.802  E-04 0.16     4.844  E-04 0.17     5.570  E-04 0.17
<   5.356  E-04 0.20     4.702  E-04 0.22     4.897  E-04 0.25     7.825  E-04 0.22
<   9.747  E-04 0.21     7.675  E-04 0.28     6.721  E-04 0.25     8.200  E-04 0.28
<   6.675  E-04 0.32     2.673  E-04 0.55     3.910  E-04 0.44     1.256  E-04 0.63
---
>   1.028  E-06 0.12     5.687  E-07 0.13     7.942  E-07 0.12     1.138  E-06 0.13
>   1.655  E-06 0.13     2.378  E-06 0.13     3.440  E-06 0.13     5.118  E-06 0.13
>   7.493  E-06 0.13     1.096  E-05 0.13     1.612  E-05 0.12     2.347  E-05 0.13
>   3.545  E-05 0.13     5.164  E-05 0.13     7.569  E-05 0.13     1.107  E-04 0.13
>   1.561  E-04 0.13     2.229  E-04 0.12     3.063  E-04 0.12     3.836  E-04 0.13
>   4.654  E-04 0.14     5.228  E-04 0.15     5.291  E-04 0.17     5.342  E-04 0.17
>   5.031  E-04 0.19     4.301  E-04 0.22     4.541  E-04 0.27     6.688  E-04 0.24
>   8.943  E-04 0.25     7.284  E-04 0.28     6.605  E-04 0.24     7.073  E-04 0.30
>   5.204  E-04 0.36     1.851  E-04 0.67     3.971  E-04 0.43     1.387  E-04 0.68
162c162
<   0.0   E+00 0.0      1.048  E-02 0.12
---
>   0.0   E+00 0.0      9.896  E-03 0.12
164c164
<             100  1.048  E-02  1.218  E-01
---
>             100  9.896  E-03  1.226  E-01
```

But how well does it run?

Chip	Price (current eBay)	Clock Speed	Caches	Number of Threads (in current system configuration)
Xeon E5-2695 v4	~$450	2.1 GHz	L1i/d 32K L2 256K L3 46080K	1 Thread/core 18 Cores/socket 2 sockets = 36 threads
ThunderX2 CN99xx	$230	2.1 GHz	L1i/d 32K L2 256K L3 32768K	4 Threads/core 32 Cores/socket 2 sockets = 256 threads

Compare 36 threads on a Xeon chip to 36 threads on the ThunderX2 chip with the same input.

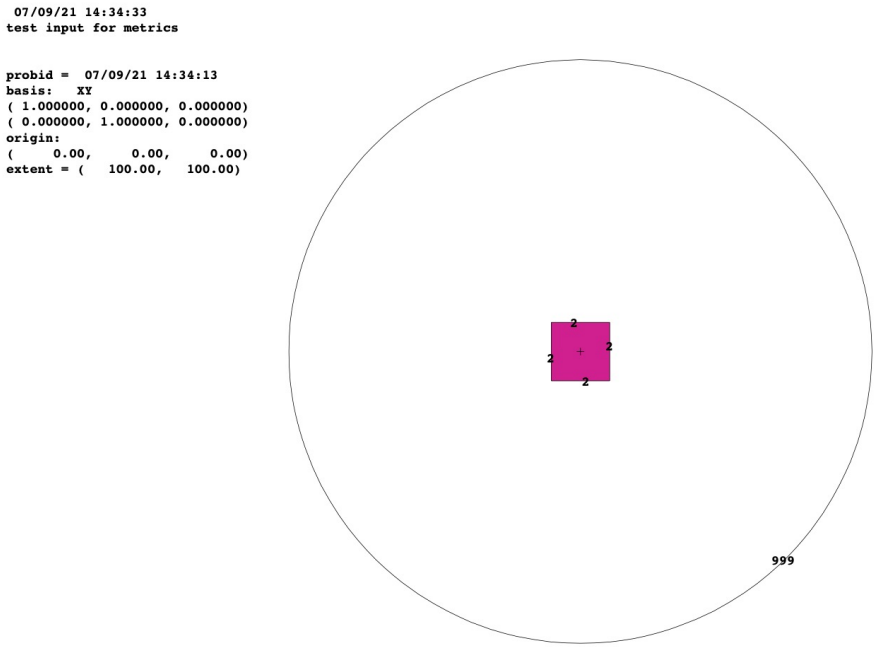
Very simple test input

test input for metrics
100 0 2 -999 imp:p=1
200 1 -6.63 -2 imp:p=1
999 0 999 imp:p=0

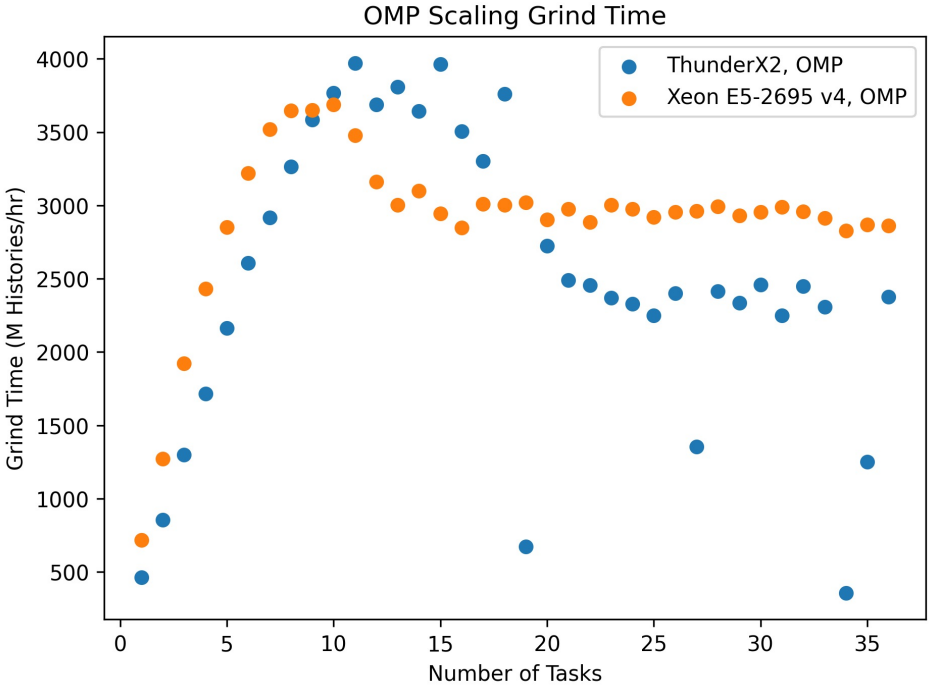
2 rpp -10 10 -10 10 -2 2
999 so 100

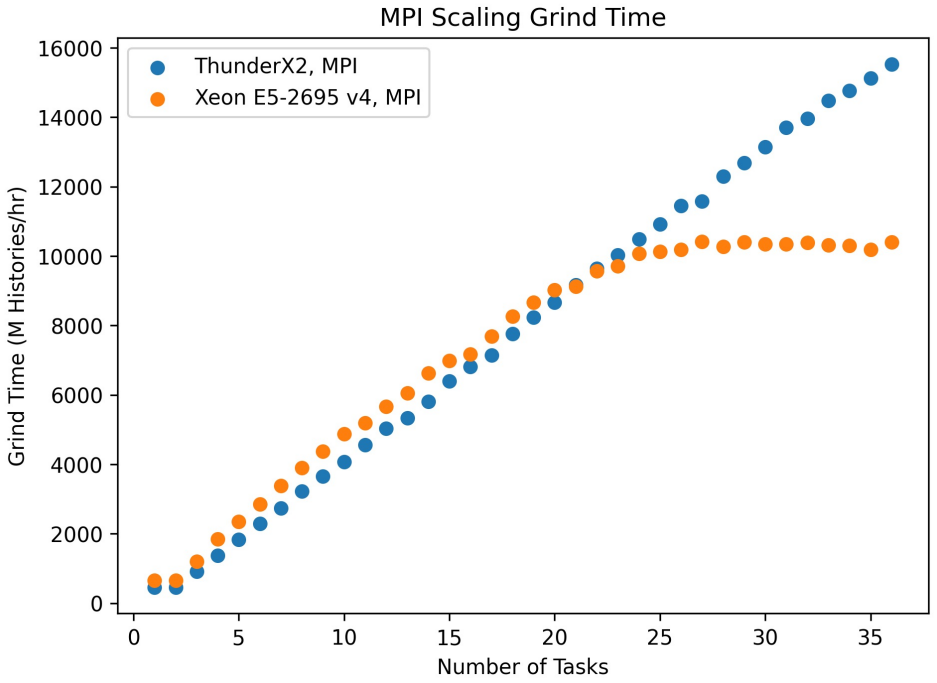
mode p
nps 1e8
sdef par=2 pos=0 0 10 erg=1.3
m1 64157 3 13027 2 31000 3 08016 12
f8:p 200
e8 0 700i 1.4

Simply a GAGG(Ce) rectangular prism in a vacuum with a photon source.



Grind Time Results





Maximum Grind time for ThunderX2 is 3971.35 M Histories/hr
For Xeon, 3687.73

ThunderX2 1.08x faster than Xeon with OMP

Maximum Grind time for ThunderX2 is 15530.47 M Histories/hr
For Xeon, 10414.27

ThunderX2 1.49x faster than Xeon with MPI

Another comparison from the NanoPi cluster:

Device	Processor	Maximum Threads	Grind Time (M Histories/hr)	Cost New (Cost Now)	Cost/M histories/hr
Dell Latitude 3570	i5-6200U @ 2.30 GHz	4	1368.8	$2500 (~$400)	1.82 (0.29)
NanoPi Fire3 Cluster	Samsung S5P6818 @ 1.4 GHz	24	2661.9	$200 (n/a)	0.075 (n/a)
ThinkPad W541	i7-4710MQ @ 2.5 GHz	8	2901.6	$2500 (~550)	0.93 (0.18)
Avery's Desktop	AMD Ryzen 5 2600 @ 3.4 GHz	12	4955.5	$1200 (n/a)	0.24 (n/a)
2019 MacBook Pro	Intel i9-9980HK @ 2.4 GHz	16	5209.56	$3,399.00 (n/a)	0.65 (n/a)

Limitations

- Off-the-shelf SBCs generally have very little RAM.
  - Fire3 cluster has 1GB per node, Raspberry Pi 4 has maximum 4GB.
- The changes to the compiler flags in the config file may lead to differences in results. This potential behavior should be investigated prior to using an ARM cluster for critical work.
- SBCs generally have slow (and small) cache and memory access times, which will decrease performance with more complex problems.
- Institutional systems like the ThunderX2 are significantly more capable, but significantly more expensive than an SBC. Total system cost may approach a Xeon system.