High Performance Computing and GPU acceleration: Where Do We Stand?

Johannes M Dieterich

jmd@FreeBSD.org
jmd@ogolem.org
jmd2@princeton.edu

November 11, 2017

Why do I care?

I am a computational chemist.

- (academic) research into chemical reactions and materials
- method development: i.e., models and algorithms
- high-performance computing developer and user

I am a FreeBSD user.

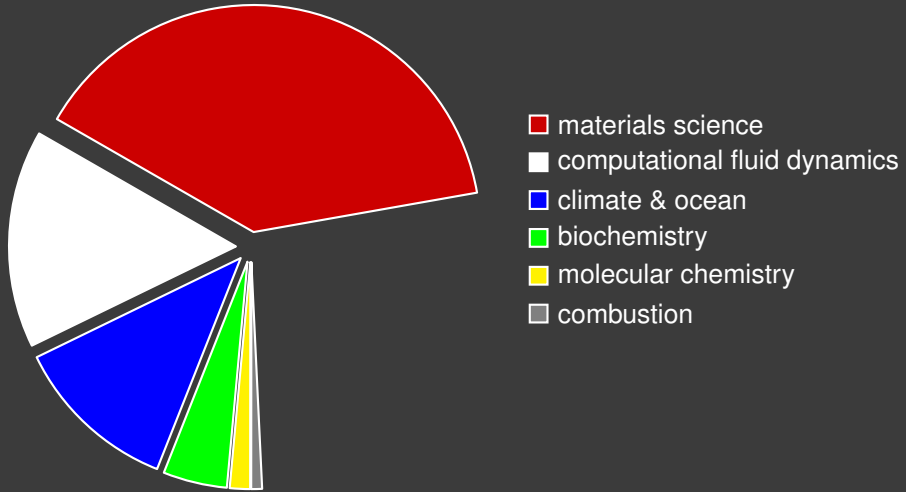
- personal hardware since 6.1-RELEASE
- development hardware since 7.1-RELEASE
- ports committer since January 2017

What is High Performance Computing (HPC)?

High Performance Computing most generally refers to the practice of aggregating computing power in a way that delivers much higher performance than one could get out of a typical desktop computer or workstation in order to solve large problems in science, engineering, or business.

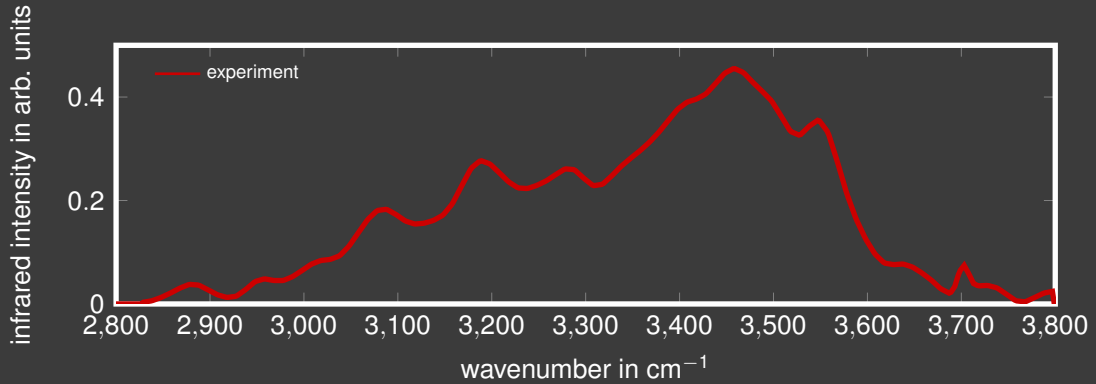
https://insidehpc.com/hpc-basic-training/what-is-hpc/

But chemistry?! HPC usage by domain



Data source: UK supercomputer ARCHER application usage of the last month, http://www.archer.ac.uk/status/codes/

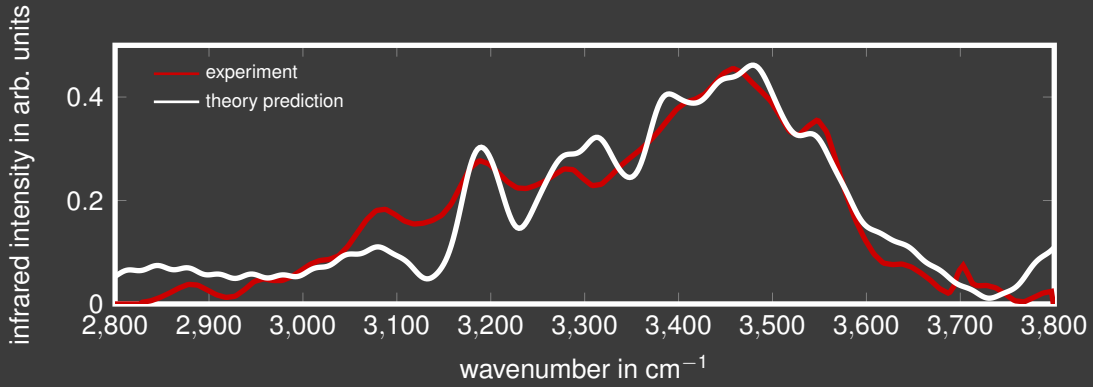
Cluster science: Experiment and Simulation



What $(H_2O)_{25}$ geometry causes this observation?

J. M. Dieterich and B. Hartke, *OGOLEM: Global cluster structure optimisation for arbitrary mixtures of flexible molecules. A multiscaling, object-oriented approach*, Mol. Phys., (2010) **108** 279; J. M. Dieterich and B. Hartke, *Observable-targeting global cluster structure optimization*, Phys. Chem. Chem. Phys., (2015) **17** 11958; J. M. Dieterich and B. Hartke, *Error-Safe, Portable, and Efficient Evolutionary Algorithms Implementation with High Scalability*, J. Chem. Theory Comput. (2016) **12** 5226.

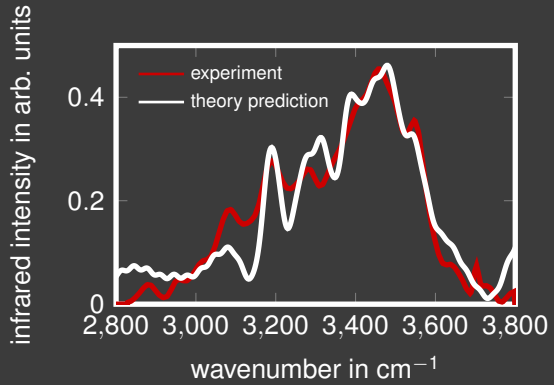
Explaining experimental data

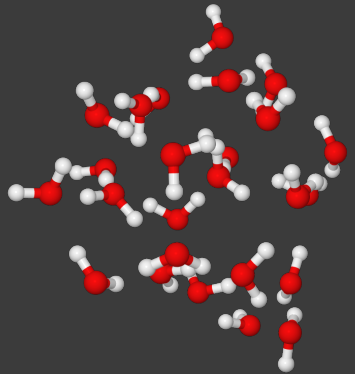


observable-targeting evolutionary algorithms global optimization, TTM3F force field

J. M. Dieterich and B. Hartke, *OGOLEM: Global cluster structure optimisation for arbitrary mixtures of flexible molecules. A multiscaling, object-oriented approach*, Mol. Phys., (2010) **108** 279; J. M. Dieterich and B. Hartke, *Observable-targeting global cluster structure optimization*, Phys. Chem. Chem. Phys., (2015) **17** 11958; J. M. Dieterich and B. Hartke, *Error-Safe, Portable, and Efficient Evolutionary Algorithms Implementation with High Scalability*, J. Chem. Theory Comput. (2016) **12** 5226.

Explaining experimental data





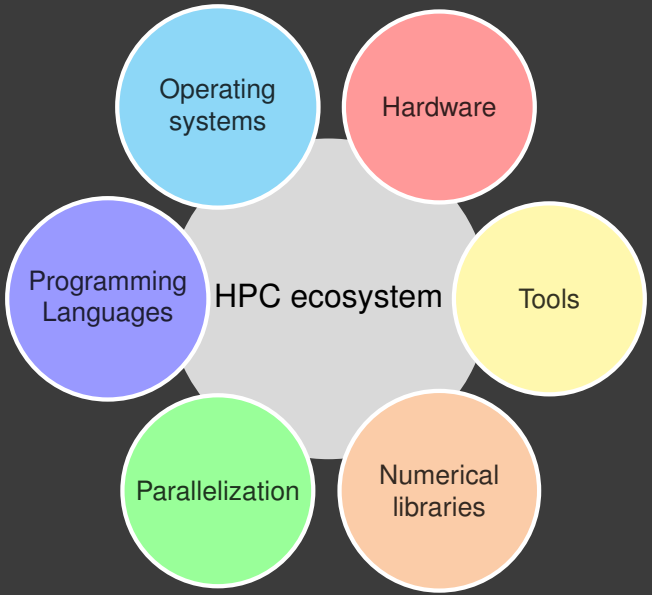
J. M. Dieterich and B. Hartke, *OGOLEM: Global cluster structure optimisation for arbitrary mixtures of flexible molecules. A multiscaling, object-oriented approach*, Mol. Phys., (2010) **108** 279; J. M. Dieterich and B. Hartke, *Observable-targeting global cluster structure optimization*, Phys. Chem. Chem. Phys., (2015) **17** 11958; J. M. Dieterich and B. Hartke, *Error-Safe, Portable, and Efficient Evolutionary Algorithms Implementation with High Scalability*, J. Chem. Theory Comput. (2016) **12** 5226.

Typical HPC installations

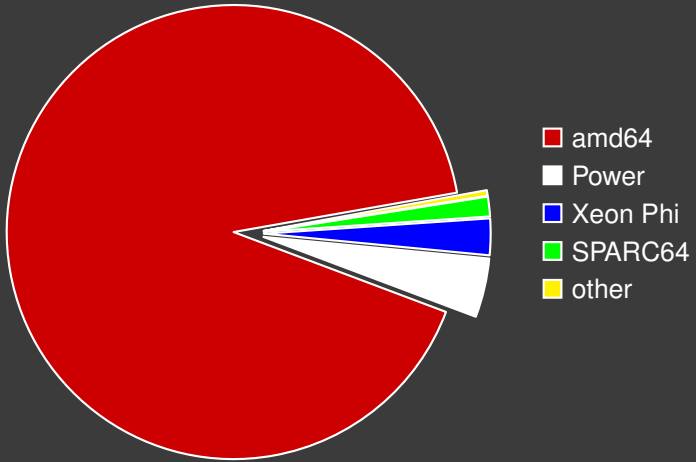
HLRN III in Berlin / Hannover, Germany

- 3552 nodes
- 85248 CPU cores
- 2.7 PFLOP
- 222 TB RAM
- 2.7 PByte storage
- 500 - 1000 kW power consumption

HLRN III *Konrad* front: https://upload.wikimedia.org/wikipedia/commons/6/62/HLRN_III_Konrad_Cray_C30_XC40.jpg

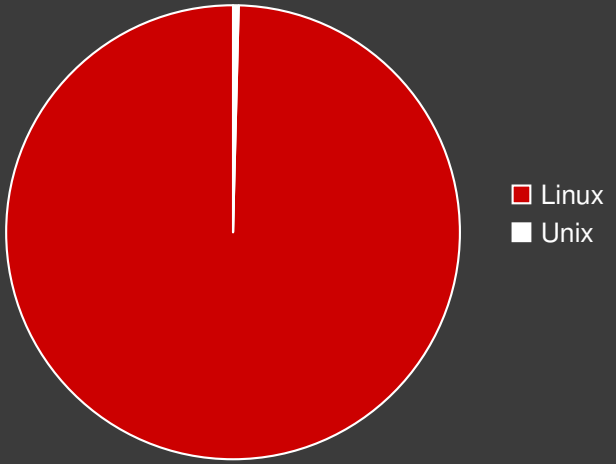

Operating systems
Hardware
Programming Languages
HPC ecosystem
Tools
Parallelization
Numerical libraries

HPC processor architectures



Data source: TOP500 list, June 2017, https://www.top500.org/statistics/list/

HPC operating systems



Data source: TOP500 list, June 2017, https://www.top500.org/statistics/list/

(Free)BSD and HPC

Strengths

- `poudriere && pkg`
  easy build and deployment of custom (optimized) packages
- documentation
- libraries and compilers setup correctly out of the box
- 31000+ ports with consistent installation and use
- traditionally excellent network and kernel performance
- some automated regression testing already in place

related: as virtualization host usage for *containered* HPC applications?

Why should we all care?

Technology and Visibility

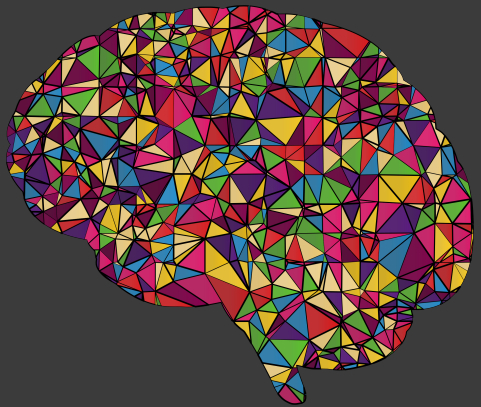
- GPU acceleration
- fundamental numerical libraries
- performance

Gears picture from https://upload.wikimedia.org/wikipedia/commons/thumb/6/6c/Cog-scripted-svg.svg/717px-Cog-scripted-svg.svg.png

Why should we all care?

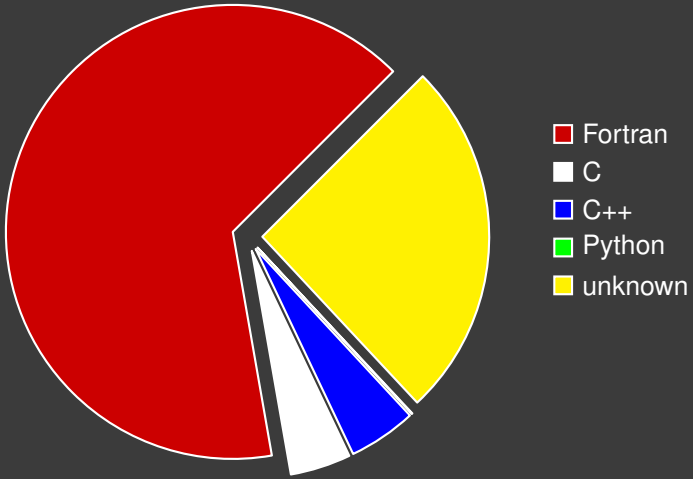
Community capital

- HPC developers
- HPC users
- new ideas



Brain picture from https://openclipart.org/detail/282980/low-poly-brain-silhouette

HPC programming languages



Data source: UK supercomputer ARCHER application usage of the last month, http://www.archer.ac.uk/status/codes/

Compilers and make environments

Compilers (on FreeBSD HEAD amd64)

- LLVM tools work very well
- `lang/gcc` has traps
  - `-Wl,-rpath=/usr/local/lib/gcc5/`
  - C++ standard library
  - OpenMP mixing
- `USES= fortran` / `compiler:openmp` goes to GCC
- Intel `icpc`, `icc` available[1]

HPC make environments

- normal makes (`gmake`, `cmake`, ...)
- homebrew makes (in `bash`, with `perl`, with `python`, ...)
- lots of OS assumptions hardcoded

[1] Intel System Studio: `https://software.intel.com/en-us/articles/intel-system-studio-2016-for-freebsd`

devel/flang: LLVM-based compiler

Features

- 64bit-only Fortran compiler (open-source)
- standards 77, 90, 95, 2003 well supported

Status

- port works for amd64
- patches not upstreamed
- based on LLVM 5.0

`r450927 (jrm)`: use `flang` by default on amd64 for `math/R`
`r452811 (jrm)`: add `USES= fortran:flang`

Flang github: `https://github.com/flang-compiler/flang`

Parallelization: inter-node

```
double globalSum = 0.0;
int err = MPI_Reduce(&sum,&globalSum,noLocalPoints,MPI_DOUBLE,MPI_SUM,0,_comm);
if(err != MPI_SUCCESS){
   MPI_Abort(_comm,err);
}
```

Message Passing Interface (MPI)

- message-based (sync/async)
- high latency
- explicit support required
- (queue support required)
- `mpicc` / `mpif90` + magic

Support in FreeBSD

- range of ports (Open MPI, mpich)
- compiler toolchain limitations (Fortran)

Parallelization: intra-node

```
double sum = 0.0;
#pragma omp parallel for default(none) shared(grid,sum)
for(size_t x = 0; x < nSlices; ++x){
    double tmpSum = 0.0;
    for(size_t col = 0; col < nCols; ++col) {
        for(size_t row = 0; row < nRows; ++row) tmpSum += grid->at(row,col,x);
    }
    #pragma omp atomic
    sum += tmpSum;
}
```

OpenMP

- pragma-based
- low latency
- offloading capable
- `${CC} -fopenmp`

Support in FreeBSD

- no support in base, `USES= compiler:openmp` depends on `lang/gcc`
- `D6362` use `devel/llvm50` on amd64
- `D11507` add LLVM's libomp to base amd64

HPC software, libraries, and capabilities

Capabilities

- queuing systems
  `sysutils/slurm-wlm`
- network support
  - InfiniBand
  - (Intel OmniPath)

Libraries

- `math/fftw3`
  FFT routines w/ OpenMP and MPI support
- `science/libint`
  integrals for quantum chemistry
- `science/libxc`
  exchange-correlation for quantum chemistry
- `math/tblis` WIP
  tensor contraction framework
- ... much, much more ...

BLAS and LAPACK

```
C := alpha*op( A )*op( B ) + beta*C
void dgemm_(TRANSA /* op(A), */ TRANSB /* op(B) */,
            M, N, K, /* dimensions of A, B, C */
            ALPHA, A, LDA /* stride length A */,
            B, LDB /* stride length B */, BETA,
            C, LDC /* stride length C */)
```

fundamental linear algebra

- probably *the* most basic HPC dependency
- standardized interface for Fortran/C
- highly efficient ($>$ 80% peak[1])
- on Linux: vendor libraries available (Intel MKL)

operations

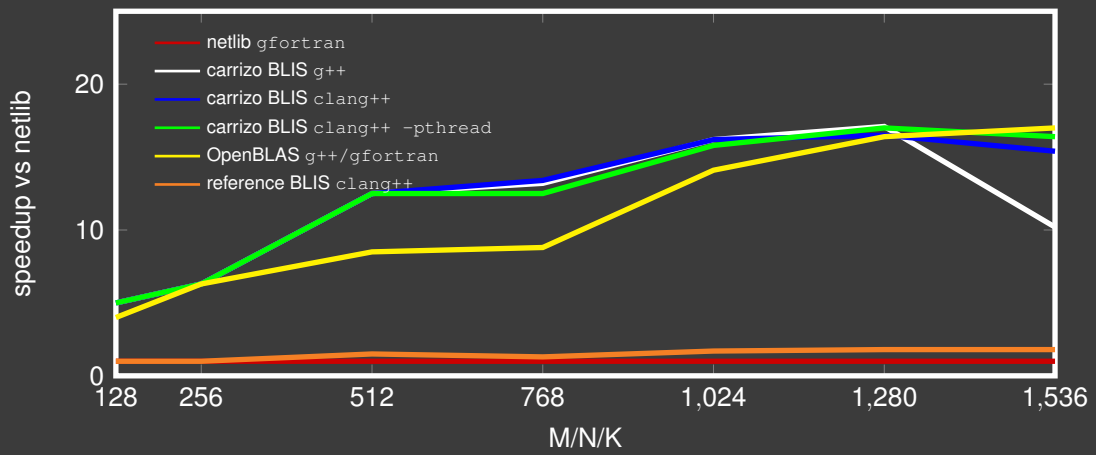
- BLAS1: $\mathcal{O}(N)$, e.g., dot product `*dot`
- BLAS2: $\mathcal{O}(N^2)$, e.g., matrix-vector product `*gemv`
- BLAS3: $\mathcal{O}(N^3)$, e.g., matrix-matrix product `*gemm`
- LAPACK: complex operations, e.g., Cholesky-decomposition `*syrk`

[1] for large BLAS3, e.g., `http://math-atlas.sourceforge.net/timing/` and `https://www.alcf.anl.gov/user-guides/bgq-dgemm-performance`

BLAS/LAPACK on FreeBSD: `Mk/Uses/blaslapack.mk`

choice	*ports*	*maintained*	*optimized*	*Fortran*	*BLAS/LAPACK*	*license*	*comments*
`netlib`	`math/blas`	yes	no	yes	yes/no	BSD3	default &
	`math/lapack`	yes	no	yes	no/yes	BSD3	reference
`atlas`	`math/atlas`		yes	yes	yes/yes	BSD3	manual build
`openblas`	`math/openblas`	yes	yes	some	yes/yes	BSD3	
`gotoblas`	`math/gotoblas`		yes	some	yes/no	BSD2	legacy
	`math/lapack`	yes	no	yes	no/yes	BSD3	
	`math/blis`	yes	yes	none	yes/no	BSD3	
	`math/libflame`		yes	none	no/yes	LGPL21	outdated

dgemm performance on FreeBSD



Data averaged over 1000 evaluations on FreeBSD HEAD @AMD A12-8800B CPU, compilation with `-O2`, all libraries compiled with `-mavx -mavx2 -msse -msse2 -msse3 -msse4 -msse4a -msse4.1 -msse4.2 -mmmx -maes -mbmi -mbmi2 -mf16c -mfsgsbase -mtune=bdver4`, BLIS compiled with `clang++` 5.0.0, netlib/OpenBLAS with `g++` 6.4.0.

Porting HPC applications to FreeBSD: Examples

TigerCI[1]

- open-source code
- C++/Fortran hybrid + OpenMP
- `cmake`
- 10s of users
- 74k SLOC
- port active

ADF suite[2]

- commercial code
- Fortran (some C/C++ bits), tcl/tk + MPI
- homebrew Python system
- 1000s of users
- 2M SLOC
- port abandoned

[1] TigerCI: `https://github.com/EACcodes/TigerCI`

[2] ADF Modeling Suite: `https://www.scm.com`

Porting HPC applications to FreeBSD

Challenges

- time / funding
- long-term viability
- Linuxisms[1]
  e.g., `#!/bin/bash`
- BSDisms
  e.g., `-Wl,-rpath=`
- port dependencies

Benefits

- portability
- compliance
- exposure of bugs

[1] Linuxisms writeup: `https://wiki.freebsd.org/AvoidingLinuxisms`

Developing on FreeBSD

One of the strengths of FreeBSD!

Debug

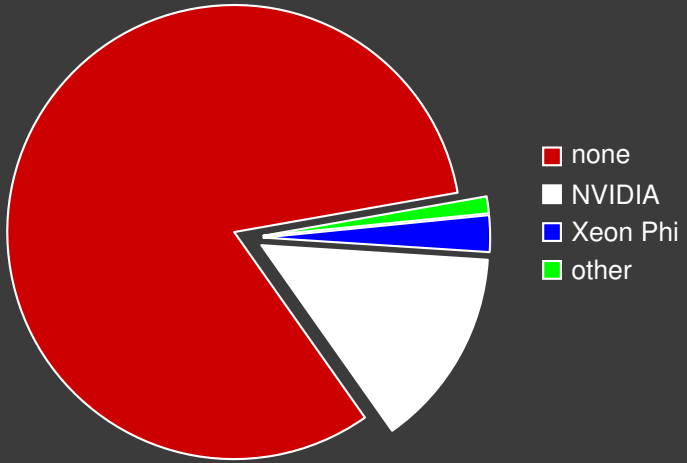
- `gdb && lldb`
- `devel/valgrind`
- ...

Profile

- `dtrace && hwpmc`
- `benchmark/flamegraph`
- `java/lightweight-java-profiler`
- ...

effectively absent: proprietary tools (`ddt`, `ifort`, ...)

HPC: accelerators/co-processors



Data source: TOP500 list, June 2017, https://www.top500.org/statistics/list/

GPU acceleration: basics

CUDA

- Fortran/C/C++, other bindings
- effectively proprietary
- NVIDIA only full implementation
  - AMD's HIP partial
  - LLVM CUDA backend misses runtime libraries
- not natively available on FreeBSD

OpenCL

- C/C++, other bindings
- open Khronos-group standard (current version: 2.2.3)
- vendor-specific OpenCL runtimes
- general runtime loader `devel/ocl-icd`
- applications are linked against `-lOpenCL`
- compute kernels are compiled at runtime

OpenCL: an example

```
__kernel void pow53PotGrid(__global double16 *data,
                           __global double16 *pot){
    const size_t idx = get_global_id(0);
    const double16 d = data[idx];
    const double16 dSq = d*d;
    const double16 d23 = cbrt(dSq);
    const double16 d53 = d23*d;
    const double pre = 5.0/3.0*2.87123400018819;
    data[idx] = d53;
    pot[idx] = pre*d23;
}

clSetKernelArg(pow53PotKernel, 0, sizeof(cl_mem), &buff);
clSetKernelArg(pow53PotKernel, 1, sizeof(cl_mem), &pot);
clEnqueueNDRangeKernel(_queue, pow53PotKernel, 1, NULL, &realPoints,
                       NULL, 0, NULL, NULL);
```

OpenCL on FreeBSD: CPU

POrtable Computing Language (POCL)

port:	`lang/pocl`
maintained:	yes (`ohartman@zedat.fu-berlin.de`)
technology:	LLVM 4.0
purpose:	CPU implementation of OpenCL easy adaptation to new targets and devices experimental CUDA backend
OpenCL standard:	1.2
current status:	crashes on Carrizo/Threadripper/Broadwell EP very latest does not package patches in development

OpenCL on FreeBSD: Intel

beignet

port:	`lang/beignet`
maintained:	yes (`x11`)
technology:	LLVM 4.0
purpose:	OpenCL runtime for integrated GPUs of Intel APUs
OpenCL standard:	1.2
current status:	double precision support only experimental

OpenCL on FreeBSD: AMD

clover

port:	`lang/clover`
maintained:	yes (`x11`)
technology:	LLVM 4.0 + Mesa 13.2.3
purpose:	OpenCL runtime for GPUs of AMD
OpenCL standard:	1.2
current status:	not supported by AMD crashes on (at least) Carrizo and Polaris

Developing OpenCL on FreeBSD

Tools

- `devel/oclgrind`
  virtual OpenCL device simulator
- `devel/cltune`
  tune OpenCL kernels for devices
- `benchmarks/clpeak`
  measure peak capabilities of OpenCL devices

Libraries

- `math/clblas`
  BLAS functions written in OpenCL (AMD)
- `math/clblast`
  tunable OpenCL BLAS library
- `math/clfft`
  FFT functions written in OpenCL (AMD)
- `math/clrng`
  uniform random number generation in OpenCL (AMD)

Other language bindings: java/aparapi

- uses Java Native Interface (JNI) to access OpenCL
- 1D kernel execution
- current port AMD version
- relies on working `libOpenCL.so`

```
class MyKernel extends com.amd.aparapi.Kernel {

    final double[] data;
    final double[] pot;

    AparapiDMatMul(...){...}

    @Override
    public void run() {
        final int i = idx = getGlobalId();
        final double d = data[idx];
        final double d23 = Math.cbrt(d*d);
        final doubel d53 = d23*d;
        final double pre = 5.0/3.0*2.87123400018819;
        data[idx] = d53;
        pot[idx] = pre*d23;
    }
}

final MyKernel ap = new MyKernel(...);
ap.setExecutionMode(Kernel.EXECUTION_MODE.GPU);
ap.execute(numberOfPoints);
```

Radeon Open Compute (ROCm) and the FreeBSDDesktop project

Fundamentals

1. `amdgpu`
   - KMS driver
   - supported by `graphics/drm-next-kmod`
2. `amdkfd`
   - compute kernel driver
   - support in `graphics/drm-next-kmod` in progress
3. `ROC Thunk`
   - usermode library interface to `amdkfd`
4. `ROC Runtime`
   - runtime libraries
5. `hcc`
   - llvm fork supporting compiling code to ROCm

Libraries/Features

- OpenCL
- HIP (CUDA emulator)
- rocBLAS
- rocFFT
- MIOpen
- hiptensorflow
- ...

Wishlist: Near-term changes

- `math/blis`
  - upstream dynamic architecture support
  - update select BLAS-only consuming ports to support BLIS
- OpenMP
  - get `D6362` into the tree
- OpenCL
  - fix `lang/pocl` and `lang/clover`?

- more people / more ports / more HPC

Wishlist: Mid-term changes

- `blaslapack:flame`
  - update `math/libflame`
  - exp-run BLIS/libflame
- OpenMP
  - `D11507` to import libomp into base for amd64
  - adjust `D6362` logic for base support
- acceleration
  - working `amdkfd` through `graphics/drm-next-kmod`
  - ROCm ecosystem into ports tree
- `devel/flang`
  - test `USES= fortran:flang` for more ports

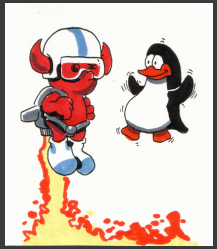
Wishlist: Long-term changes

- `blaslapack:flame`
  - default for tier 1
- OpenMP
  - support more than amd64
- acceleration
  - OpenCL via ROCm
  - be on top of `amdgpu`/`amdkfd`/ROCm releases
- `devel/flang`
  - support more than amd64
  - default for amd64
- `devel/linuxisms` port?

Cut into the 99+% Linux monoculture: Be a *used* HPC platform!

FreeBSD Acknowledgements

- Matthew `kip` Macy
- Koop `kwm` Mast
- Mark `markj` Johnston
- Johannes Lundberg
- Hans Petter `hps` Selasky
- Steve `swills` Wills
- `rene` Ladan
- Pete Wright
- FreeBSDDesktop contributors/users/testers
- Joseph `jrm` Mingrone



Please test, report, and contribute: `https://github.com/FreeBSDDesktop`

Rocket beastie: `https://www.flickr.com/photos/bsdfan/`