Porting EOSPAC6 to Sierra

Anna Pietarila Graham
David Pimentel
Sept 2020
Topics

1. EOSPACE6 Overview
2. Porting goals and strategy
3. Current status and performance
4. Summary and next steps
EOSPAC6

• Equation of state library
  • Used to access the Sesame equation-of-state data library and interpolate the data
    • Additional capabilities: mixing, smoothing
  • Development started in 1982, currently at version 6 (2001)
  • Written in C, called from Fortran & C client codes
• Deployed on all LANL HPC systems: 5 compilers, 16 architectures (excluding classified systems)
• Used by multiple large LANL physics codes
• Testing and verification
  • Nightly testing on multiple platforms
  • ~450 regression tests
(Simplified) client code use of EOSPAC6

Set up phase:
Create & load tables

*\(x\), *\(y\), *\(F\), *\(\text{derivatives}\)

Calculation (interpolation) phase

*\(x\), *\(y\), *\(F\), *\(\text{derivatives}\)

Clean up phase
(Simplified) client code use of EOSPAC6

Set up phase:
Create & load tables

*\(x\), *\(y\), *\(F\), *derivatives

Calculation (interpolation) phase

*\(x\), *\(y\), *\(F\), *derivatives

Clean up phase
Porting strategy

• From client code point of view:
  • Goal: Most physics computations ported to GPU
  • EOSPAC calls embedded in other code
  • → relevant data already on the GPU

  →Work with device pointers as much as possible
  • Minimize data transfers between host and target
(Simplified) client code use of EOSPAC6

Set up phase:
Create & load tables

Calculation (interpolation) phase
*x, *y, *F, *derivatives

Clean up phase

Eos_GpuOffloadData
Offload Sesame tables to GPU
Disable further changes to tables
Set internal flag

*x, *y, *F, *derivatives
Porting strategy

OpenMP offloading

1) Portable & easy to maintain
2) No offload specific kernels need to be written

```c
#ifdef DO_OFFLOAD
#pragma omp target if(useGpuData) is_device_ptr(var)
{
#pragma omp teams distribute parallel for
#endif /* DO_OFFLOAD */
```

3) Verification with client code: Shaped charge test setup from Pagosa
   - Representative of typical code path used by client code
   - Each client code MPI rank has its own instance of EOSPAC
Current status

• Refactored code e.g.
  • Refactor subroutines to do multiple points instead of single point
  • Rewrite of extrapolated data section
  • Use flat arrays instead of multi-dimensional arrays:
    EOS_REAL *F_flat = !useGpUData?&F[0][0]:&F[0];

• Ported code path used by Pagosa regression test
  • Interpolation & search routines
  • Misc. bits: unit conversion etc.

• Offloaded code passes tests: EOSPAC6 and Pagosa regression tests
Performance of eos_Interpolate

API called by client code:

```c
void eos_Interpolate
(EOS_INTEGER *tableHandle, EOS_INTEGER *nXYPairs,
 EOS_REAL *xVals, EOS_REAL *yVals,
 EOS_REAL *fVals, EOS_REAL *dFx, EOS_REAL *dFy,
 EOS_INTEGER *errorCode);
```

interpolation & search, bookkeeping, unit conversion, extrapolation checking (not ported)
Performance: **CPU** release vs **offload**

- **Power9**
  - CPU: serial
  - GPU: v100
- **IBM XL**
  - OpenMP 4.5
- **Flags:**
  - `-O` `-qfloat=nomaf`
  - `-qstrict=precision`
- **GPU:**
  - `-qoffload` `-qsmp=noopt`

![Graph showing performance comparison between CPU and GPU](image)
Performance: add **CPU w/o extrapolation checking**
Performance: add optimized offload

CPU w/o extrap. checks

CPU

GPU

GPU -qsmp=omp

GPU:
-qoffload –qsmp=omp
Performance

RUNTIME RATIO

GPU opt/CPU wo extrap.check

GPU qsmp= noopt/CPU wo extrap.check

GPU faster

x135 speed up

CPU faster

NXYPairs
**EOSAPC6 GPU memory usage** (single Sesame table)

- **EOSAPC6 mem usage**
- **EOSAPC input&output (allocated by client code)**
- **GPU (v100) avail mem**

![Graph showing EOSAPC6 GPU memory usage](image-url)
Performance bottleneck: extrapolation warnings, error checks (on CPU)

66.79% 313.691s  eos_Interpolate
66.79% 313.691s  | eos_InterpolateEosInterpolation
64.18% 301.421s  | eos_InterpolateRecordType1
64.18% 301.421  | | _eos_InterpolateRecordType1
37.73% 177.2s  | | eos_CheckExtrapRecordType1_using_extrapolationBounds
16.72% 78.5102s  | | eos_RationalInterpolateXY
10.49% 49.2501s  | | | _eos_srchdf
8.35% 39.2101s  | | | | eos_GetStandardErrorCodeFromCustomErrorCode
4.31% 20.25s  | | | | eos_BiRationalInterpolate
8.84% 41.5201s  | | | | eos_RationalInterpolate
5.35% 25.1501s  | | | | _eos_srchdf
4.28% 20.09s  | | | | eos_GetStandardErrorCodeFromCustomErrorCode
3.09% 14.52s  | | | | eos_RationalInterpolate4

237s of 314s (75%) spent in error and warning code checks
Summary

• Completed proof of concept version using OpenMP offloading
  • Passes validation tests
  • CPU-GPU break-even point: >2. $\times 10^5$ points
  • 135X speed-up when nXYPoints > 10 million

• Identified performance bottle neck in extrapolation checks
Future work

• Port rest of code & create an alpha-release of GPU-enabled library

• Address extrapolation checking
  • New user option to disable checking unless API eos_CheckExtrap is called from client code

• Further performance improvements
  • Temporary array memory usage
  • Optimization of individual kernels
  • …
Over 70 years at the forefront of supercomputing