Building Legion and Exploring Legion Runtime

Daniel M. Topa

April 25, 2025

Overview

This document summarizes the contents of the Legion 'examples/' directory, grouped by function and pedagogical purpose. It also includes a shell script to compile and run a subset of these examples on an 8-core system, verifying a successful installation of the Legion runtime. It concludes with a discussion on running Legion on various backends and a complete 'spack info legion' capture.

Running Legion on CPUs and GPUs

Legion can execute on a variety of backends:

- **CPU-only:** Use `-ll:cpu N` to allocate N logical CPU processors. Most examples run successfully in this mode.
- **GPU-only:** Use `-ll:gpu N` to target N GPUs. This requires building Legion with `+cuda` and setting a valid `cuda_arch`.
- **Hybrid CPU + GPU:** You can specify both `-ll:cpu` and `-ll:gpu` options to enable heterogeneous execution.

To inspect the runtime configuration at launch, include `-hl:show_rdetail` as a command-line argument.

Building Legion with Spack

Spack provides a reliable method to install the Legion runtime and its dependencies. To gain access to the Legion examples and auxiliary tools, we use the `--keep-stage` option:

Installation and Staging

```
spack install --keep-stage legion +cuda +fortran +openmp +hwloc +papi +python network=gasnet
spack stage legion
spack cd legion
ls
apps/ CMakeLists.txt Dockerfile language/ README.perf.md test/ VERSION
bindings/ deprecated/ doxygen/ LICENSE.txt README.test.md test.py
CHANGES.txt doc/ examples/ perf.py realm/ tools/
cmake/ docker/ jupyter_notebook/ README.md runtime/ tutorial/
```

The `examples/` directory contains runnable programs used to validate your installation.

Spack Info Output

```
  \$ spack info legion
==> Warning: The packages:all:compiler preference has been deprecated in Spack v1.0, and is currently ignored. It will be removed from config in Spack v1.2.
CMakePackage: legion

Description:
    Legion is a data-centric parallel programming system for writing
    portable high performance programs targeted at distributed heterogeneous
    architectures. Legion presents abstractions which allow programmers to
    describe properties of program data (e.g. independence, locality). By
    making the Legion programming system aware of the structure of program
    data, it can automate many of the tedious tasks programmers currently
    face, including correctly extracting task- and data-level parallelism
    and moving data around complex memory hierarchies. A novel mapping
    interface provides explicit programmer controlled placement of data in
    the memory hierarchy and assignment of tasks to processors in a way that
    is orthogonal to correctness, thereby enabling easy porting and tuning
    of Legion applications to new architectures.

Homepage: https://legion.stanford.edu/

Preferred version:
    25.03.0       [git] https://github.com/StanfordLegion/legion.git at commit 04716e3b3686d4af71e6a4398dfbe8cd869c057b

Safe versions:
    master        [git] https://github.com/StanfordLegion/legion.git on branch master
    stable        [git] https://github.com/StanfordLegion/legion.git on branch stable
    25.03.0       [git] https://github.com/StanfordLegion/legion.git at commit 04716e3b3686d4af71e6a4398dfbe8cd869c057b
        ...

Deprecated versions:
    23.03.0       [git] https://github.com/StanfordLegion/legion.git at commit 12f6051c9d75229d00ac0b31d6be1ff2014f7e6a
    22.12.0       [git] https://github.com/StanfordLegion/legion.git at commit 9ed6f4d6b579c4f17e0298462e89548a4f0ed6e5
        ...

Variants:
    bindings [false]                     false, true
        Build runtime language bindings (excl. Fortran).
    bounds_checks [false]                false, true
        Enable bounds checking in Legion accessors.
    build_system [cmake]                 cmake
        Build systems supported by the package
    cuda [false]                         false, true
        Enable CUDA support.
    cuda_arch [70]                       10, 100, 100a, 101, 101a, 11, 12, 120, 120a, 13, 20, 21, 30, 32, 35, 37, 50, 52, 53, 60, 61, 62, 70, 72, 75, 80, 86, 87, 89, 90, 90a
        GPU/CUDA architecture to build for.
    cuda_hijack [false]                  false, true
        Hijack application calls into the CUDA runtime (+cuda).
    cuda_unsupported_compiler [false]    false, true
        Disable nvcc version check (--allow-unsupported-compiler).
    cxxstd [17]                          11, 14, 17, 20
        C++ standard
    fortran [false]                      false, true
        Enable Fortran bindings.
    gc [false]                           false, true
        Enable garbage collector logging
    hdf5 [false]                         false, true
        Enable support for HDF5.
    hwloc [false]                        false, true
        Use hwloc for topology awareness.
    kokkos [false]                       false, true
```

```
        Enable support for interoperability with Kokkos.
    libdl [true]                        false, true
        Enable support for dynamic object/library loading.
    max_dims [3]                        none
        Set max number of dimensions for logical regions.
    max_fields [512]                    none
        Maximum number of fields allowed in a logical region.
    max_num_nodes [1024]                none
        Maximum number of nodes supported by Legion.
    network [none]                      none, gasnet, mpi, ucx
        The network communications/transport layer to use.
    openmp [false]                      false, true
        Enable support for OpenMP within Legion tasks.
    output_level [warning]              none, debug, error, fatal, info, print, spew, warning
        Set the compile-time logging level.
    papi [false]                        false, true
        Enable PAPI performance measurements.
    privilege_checks [false]            false, true
        Enable runtime prividge checks in Legion accessors.
    prof [false]                        false, true
        Install Rust Legion prof
    python [false]                      false, true
        Enable Python support.
    redop_complex [false]               false, true
        Use reduction operators for complex types.
    redop_half [false]                  false, true
        Use reduction operators for half precision types.
    rocm [false]                        false, true
        Enable ROCm support
    shared [false]                      false, true
        Build shared libraries.
    spy [false]                         false, true
        Enable detailed logging for Legion Spy debugging.
    sysomp [false]                      false, true
        Use system OpenMP implementation instead of Realm\textquotesingle s
    zlib [true]                         false, true
        Enable zlib support.

    when +rocm
      amdgpu_target [none]              none, gfx1010, gfx1011, gfx1012, gfx1013, gfx1030, gfx1031, gfx1032,
           gfx1033, gfx1034, gfx1035, gfx1036, gfx1100, gfx1101, gfx1102, gfx1103, gfx701,
                                        gfx801, gfx802, gfx803, gfx900, gfx900:xnack-, gfx902, gfx904, gfx906,
                                             gfx906:xnack-, gfx908, gfx908:xnack-, gfx909, gfx90a, gfx90a:
                                            xnack+,
                                        gfx90a:xnack-, gfx90c, gfx940, gfx941, gfx942
          AMD GPU architecture
      hip_hijack [false]                false, true
          Hijack application calls into the HIP runtime
      hip_target [ROCM]                 CUDA, ROCM
          API used by HIP

    when build_system=cmake
      build_type [Release]              Debug, MinSizeRel, RelWithDebInfo, Release
          CMake build type
      generator [make]                  none
          the build system generator to use

    when build_system=cmake ^cmake@3.9:
      ipo [false]                       false, true
          CMake interprocedural optimization

    when network=gasnet
      conduit [none]                    none, aries, ibv, mpi, ofi-slingshot11, ucx, udp
          The GASNet conduit(s) to enable.
      gasnet_debug [false]              false, true
          Build gasnet with debugging enabled.
      gasnet_root [none]                none
          Path to a pre-installed version of GASNet (prefix directory).
```

```
126
127 Build Dependencies:
128     c      cuda  fortran hdf5  hsa-rocr-dev kokkos    llvm-amdgpu ninja py-cffi  py-pip        python ucc  zlib-
        api
129     cmake  cxx   gmake   hip   hwloc        libfabric mpi         papi   py-numpy py-setuptools rust   ucx
130
131 Link Dependencies:
132     cuda  hdf5  hip  hsa-rocr-dev hwloc kokkos libfabric llvm-amdgpu mpi papi  py-cffi py-numpy python ucc  ucx
        zlib-api
133
134 Run Dependencies:
135     None
136
137 Licenses:
138     Apache-2.0
```

Validation Script for 8-Core Run

The following script builds and tests a handful of Legion examples using 8 logical CPU cores:

```
1  #!/bin/bash
2
3  # Set to Legion source root if not already in path
4  LEGION_DIR=~/src/legion/examples
5  cd "$LEGION_DIR" || { echo "Legion␣examples␣not␣found."; exit 1; }
6
7  EXAMPLES=(
8    inline_tasks
9    allreduce
10   concurrent_tasks
11   attach_array_daxpy
12   ghost
13   layout_constraints
14   spmd_cgsolver
15   circuit
16 )
17
18 for ex in "${EXAMPLES[@]}"; do
19   echo "\n=====␣Building␣$ex␣====="
20   cd "$LEGION_DIR/$ex" || continue
21   make clean && make -j8
22   echo "\n=====␣Running␣$ex␣with␣8␣CPUs␣====="
23   ./$ex -ll:cpu 8
24   echo "\n=====␣Done␣$ex␣====="
25 done
```