LCA14-412: GPGPU on ARM SoC

Thu 6 March, 2.00pm, T.Gall, G.Pitney

Agenda

Shamrock - Gil Pitney

sqlite accelerated with OpenCL - Tom Gall

GPGPU Goals

- Recognizing that:
  - GPUs are much more energy efficient for highly parallel computing problems than using just the main CPU.
  - Big performance gains (> 20X) can be had using the GPU for certain open source workloads.
- Decided:
  - to get hardware and start gaining some experience with GPGPU drivers;
  - to accelerate some OSS projects with GPGPU; and
  - to get an OSS OpenCL project started for CPU-only (ARM-only) kernels
- A CPU-only OpenCL implementation is valuable...
  - as a fallback implementation where a GPGPU driver doesn't exist;
  - as a means to debug OpenCL kernels (where GPGPU debuggers don't exist); and
  - as a means to prove MCJIT/ARM capability, which may prove useful for other projects relying on LLVM (eg: Android).

Shamrock: a type of Clover

- GPGPU team evaluated OSS OpenCL implementations
  - pocl vs SNU vs clover: Details in GPGPU whitepaper (WIP).
- We chose clover for the non-GPGPU OpenCL solution:
  - clover calls LLVM JIT lib APIs directly, whereas pocl did an exec().
  - clover appeared well architected, and easier adapt to new H/W.
  - TI (a Linaro member company) had successfully ported clover to Keystone II ARM+DSP platform, so potential to profit from the generic updates.
- Goals:
  - Get clover working on ARM (Chromebook, ODROID-XU)
  - Update LLVM dependencies: (clover was on LLVM 3.0 / x86 !)
  - Port and run Khronos conformance tests on ARM/clover.
  - Update to OpenCL version 1.2, then 2.0.
- Pulled into git.linaro.org and renamed “shamrock”
  - https://git.linaro.org/gpgpu/shamrock.git

Shamrock: port to ARM, newer LLVM

- Port to ARM from x86.
  - Fortunately, clover used CMake as the portable Makefile generator/build system.
  - This bit of host code ensures the right ARM target for LLVM is used:
    - target_opts.Triple = **llvm::sys::getDefaultTargetTriple()**;
- Port to newer LLVM versions.
  - Moved to LLVM 3.2, then v3.3, then finally trunk (3.5svn).
  - Each new LLVM version change involves handling one or more of the following (often undocumented changes):
    - APIs moved into other or new libraries;
    - API name or signature changes;
    - Deprecated features;
    - Rearranged header files;
  - Lot's of time spent finding the right LLVM configuration, and rebuilding LLVM/clang.
  - But, overall, LLVM/clang functionality remained pretty stable.

Shamrock: Adding MCJIT support (1 / 3)

- LLVM JIT engine being replaced by new MCJIT engine.
  - clover was based on old JIT engine in LLVM 3.0.
  - Found that old JIT emitted some invalid instructions for ARMv7.
  - Also, MCJIT engine doesn't support "lazy compilation", a feature relied upon by clover.
  - So, to get OpenCL kernels to compile and run on ARM, needed to move to MCJIT, and solve lazy compilation issue.
- How was it done?
  - Lots of help from Kaliedoscope/MCJIT tutorial:
    - Using MCJIT with the Kaleidoscope Tutorial
  - RTTI Constraint:
    - Shamrock uses Boost, which requires RTTI.
    - To get MCJIT's MemoryManager class to link with shamrock, needed to enable RTTI to enable linking with with shamrocks' MemoryManager subclass.

```
% CC=gcc CXX=g++ ./configure --prefix=/opt/llvm --enable-jit --enable-targets=arm --enable-optimized --
enable-assertions --with-float=hard --with-abi=aapcs-vfp
% make -j4 REQUIRES_RTTI=1
```

Shamrock: Adding MCJIT support (2 / 3)

- Issue: Shamrock registered a "LazyFunctionCreator" callback with JIT engine to allow linking in unresolved symbols defined in the host CPU shamrock library:

```
p_jit->DisableSymbolSearching(true);
p_jit->InstallLazyFunctionCreator(&getBuiltin);
```

- where,

```
void *getBuiltin(const std::string &name)
{
        if (name == "get_global_id")
        return (void *)&get_global_id;
        else if (name == "get_work_dim")
        return (void *)&get_work_dim;
        // etc…
}
```

- But MCJIT no longer supports that callback feature.
- So, overrode MCJIT's **LinkingMemoryManager::getSymbolAddress()** to allow kernels to link with shamrock-defined OpenCL builtins…

Shamrock: Adding MCJIT support (3 / 3)

```
// Create a custom memory manager for MCJIT
class ClientMemoryManager : public SectionMemoryManager
{
   //  [...]
public:
  /// This method returns the (host) address of the specified function.
  virtual uint64_t getSymbolAddress(const std::string &Name);
};

uint64_t ClientMemoryManager::getSymbolAddress(const std::string &Name)
{
  uint64_t addr = (uint64_t)getBuiltin(Name);
  if (!addr)
     report_fatal_error("Program used external function '" + Name +
                        "' which could not be resolved!");
  return addr;
 }

 p_jit = llvm::EngineBuilder(p_module)
                        .setErrorStr(&err)
                        .setUseMCJIT(true)
                        .setMCJITMemoryManager(new ClientMemoryManager())
```

Shamrock: Status, Next Steps

- Status:
  - Shamrock == clover + LLVM 3.5svn + MCJIT: https://git.linaro.org/gpgpu/shamrock.git/shortlog/refs/heads/mcjit
  - Passes original clover sanity tests (except some builtins and native kernel) on both ARM v7 (Chromebook, ODROID-XU) and x86_64 (Ubuntu VirtualBox VM).
  - Ported Khronos OpenCL v1.1 conformance test suite to ARM.
- Next Steps:
  - Merge with TI's opencl git repo, to profit from cleanup and bug fixes already made.
  - Run Khronos OpenCL v1.1 Conformance tests on ARM:
    [Results TBD].
  - Make fixes for shamrock based on test suite run.
  - Update to OpenCL v 1.2, then v 2.0

Sqlite Background

- A very popular “embedded database”
  - uses SQL for query, has c api
  - able to be compiled into your application
  - very friendly license
- Android, iOS
- Popular uses
  - ngix + foo + sqlite
  - firefox
  - chrome
- Why?
  - Used in many popular applications
  - Useful testcase for GPGPU on ARM

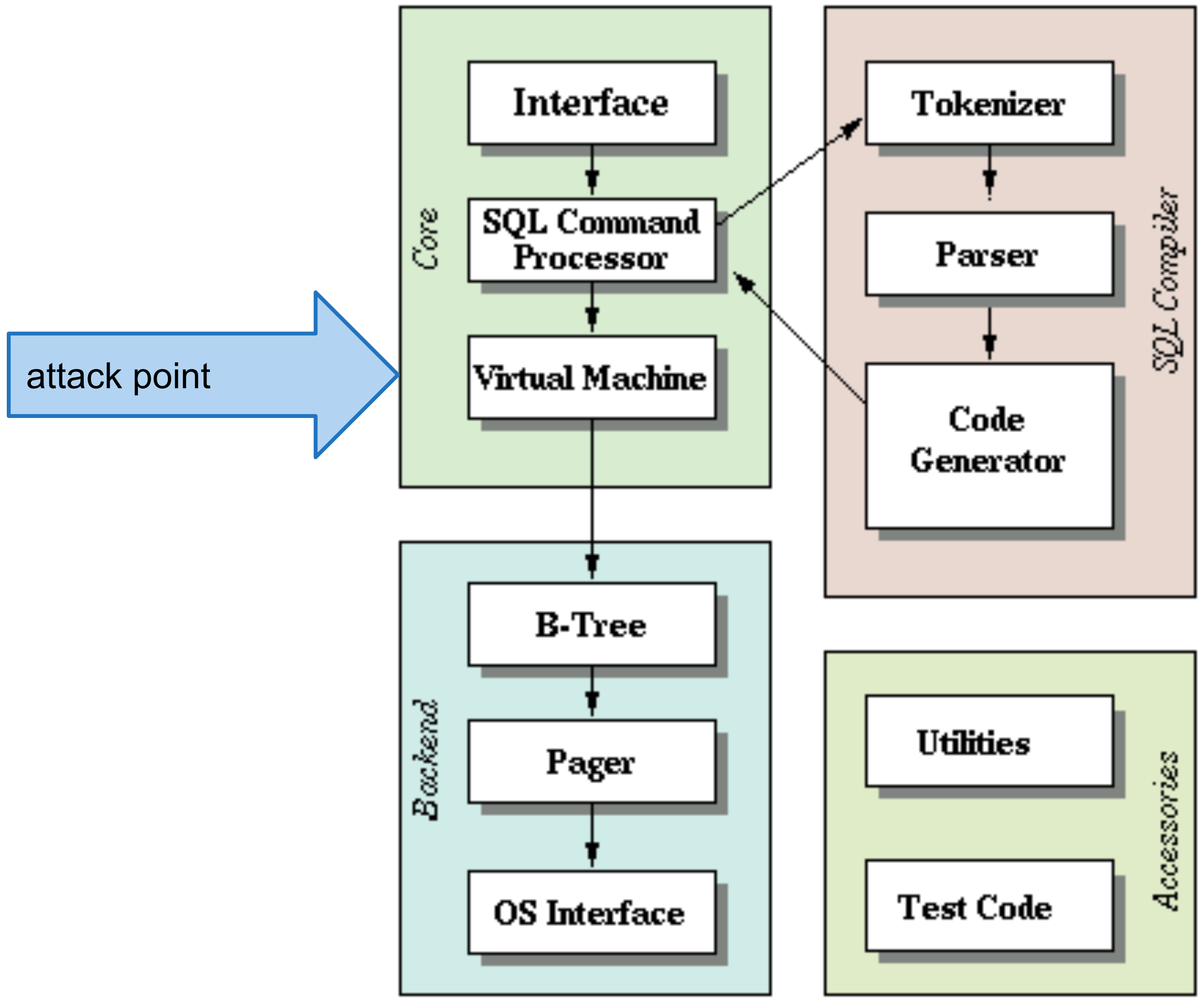
Sqlite Initial Look At Performance

- Chromebook
  - dual A15 1.7Gz
  - Mali T604
- 100,000 entry database
  - 1 table
    - 7 columns, 1 primary key (int), 3 ints, 3 floats
  - select * from test
    - ~.420 seconds
    - no sorting
    - no math operations
  - obtain count of how many rows in table
    - ~14-16 milliseconds

Sqlite : what's perf think?

```
perf record sq-cl
 26.87%  sq-cl  sq-cl                [.] sqlite3VdbeExec
 11.13%  sq-cl  libpthread-2.17.so  [.]
__pthread_mutex_unlock_usercnt
  9.79%  sq-cl  [kernel.kallsyms]   [k] 0x8010b9b4
  7.49%  sq-cl  libpthread-2.17.so  [.] pthread_mutex_lock
  5.18%  sq-cl  sq-cl               [.] columnMem
  2.88%  sq-cl  sq-cl               [.] columnMallocFailure
  2.69%  sq-cl  sq-cl               [.] sqlite3BtreeCursorHasMoved
  2.30%  sq-cl  sq-cl               [.] sqlite3VdbeMemStoreType
  1.73%  sq-cl  sq-cl               [.] sqlite3_mutex_leave
  1.73%  sq-cl  sq-cl               [.] sqlite3VdbeMemNulTerminate
  1.73%  sq-cl  sq-cl               [.] sqlite3VdbeCursorMoveto
  1.54%  sq-cl  sq-cl               [.] sqlite3VdbeSerialGet
```

Sqlite Architecture



Approach

Random

tune for data organized

copy in / copy out when using OpenCL kernels

Modification of / Extension of VM allows to minimize changes.

Not all operations make sense in OpenCL

Sqlite OpenCL Status

In progress.

More about Linaro Connect: http://connect.linaro.org
More about Linaro: http://www.linaro.org/about/
More about Linaro engineering: http://www.linaro.org/engineering/
Linaro members: www.linaro.org/members