
SIGGRAPH2015
Xroads of Discovery

SIGGRAPH2015
Xroads of Discovery
The 42nd International Conference and Exhibition
on Computer Graphics and Interactive Techniques

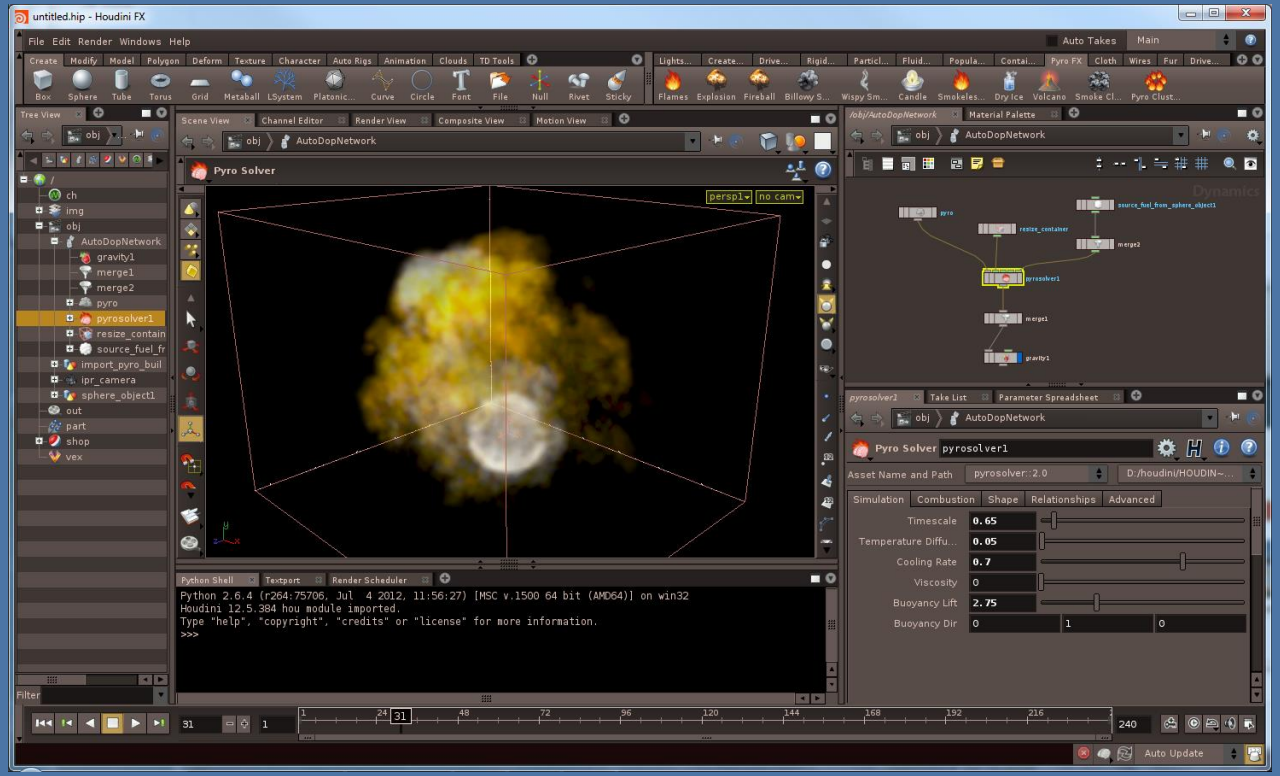
Multithreading in Visual Effects

Jeff Lait
Side Effects Software Inc.

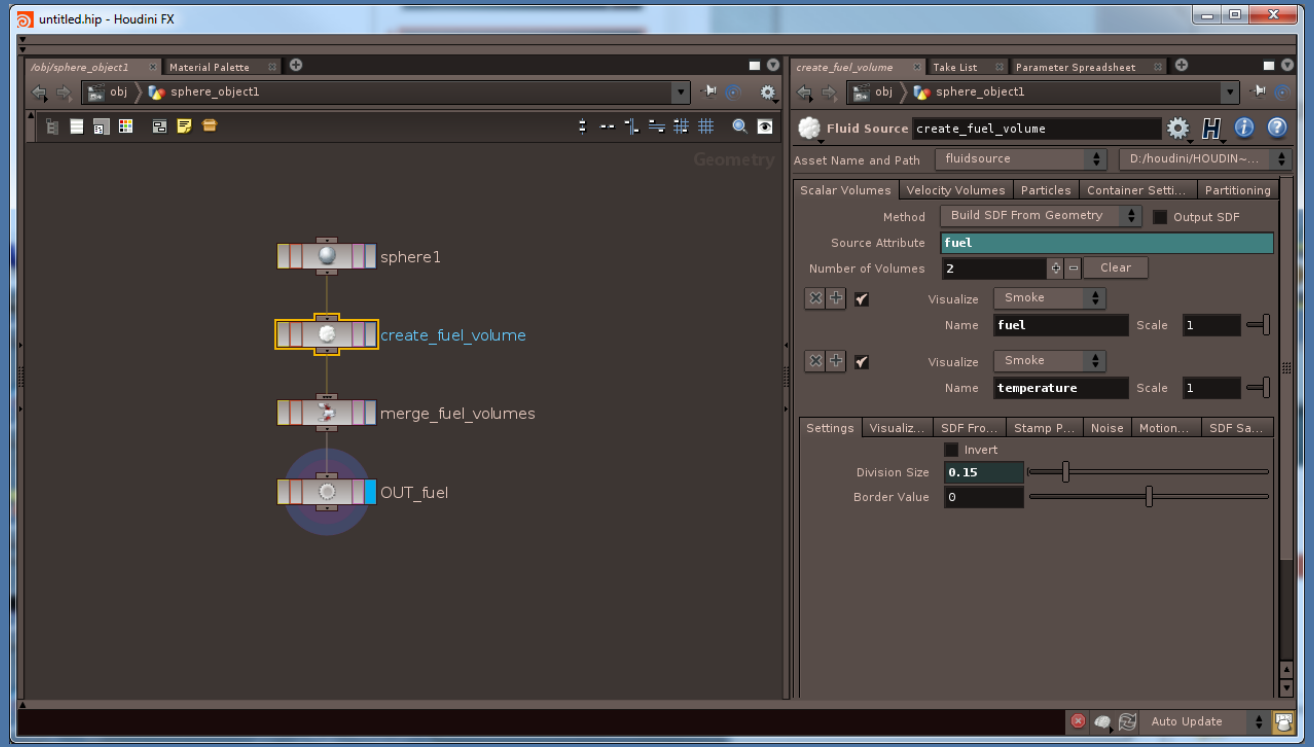
Multithreading in Houdini

Definitions

Houdini


untitled.hip - Houdini FX
File Edit Render Windows Help
Pyro Solver
AutoDopNetwork
Python 2.6.4 (r264:75706, Jul 4 2012, 11:56:27) [MSC v.1500 64 bit (AMD64)] on win32
Houdini 12.5.384 hou module imported.
Type "help", "copyright", "credits" or "license" for more information.
>>>
Timescale
Temperature Diffu...
Cooling Rate
Viscosity
Buoyancy Lift
Buoyancy Dir

What are Nodes?



Mantra

VEX

- Our SIMD shading language
- Superficially similar to Renderman Shading Language or GLSL
  - But is not just used for shading!
  - Deformation, Simulation, Procedurals, etc..

Multithreading in Houdini

Distribution

Distribution

- Why thread when you can run multiple processes?
  - Usual excuse for GIL in Python
- We already all distribute: Renderfarms

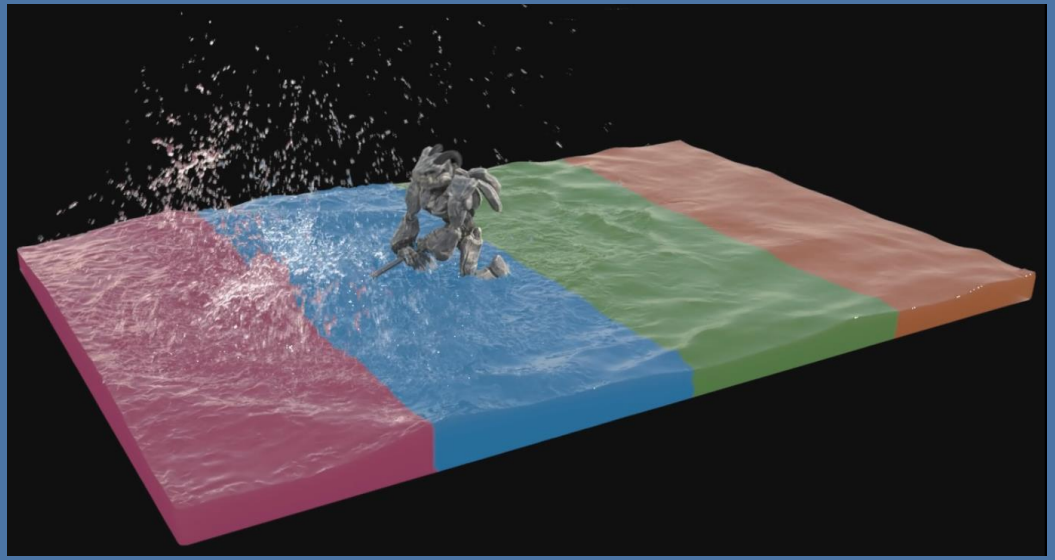
Only One Mouse

Distribution

- **"All programming is an exercise in caching.”**
  - Terje Mathisen
- Difficulty is sharing cached structures
- CPU architectures hardware-accelerate sharing

Distribution

- Distribution forces data locality



Multithreading in Houdini

Challenges

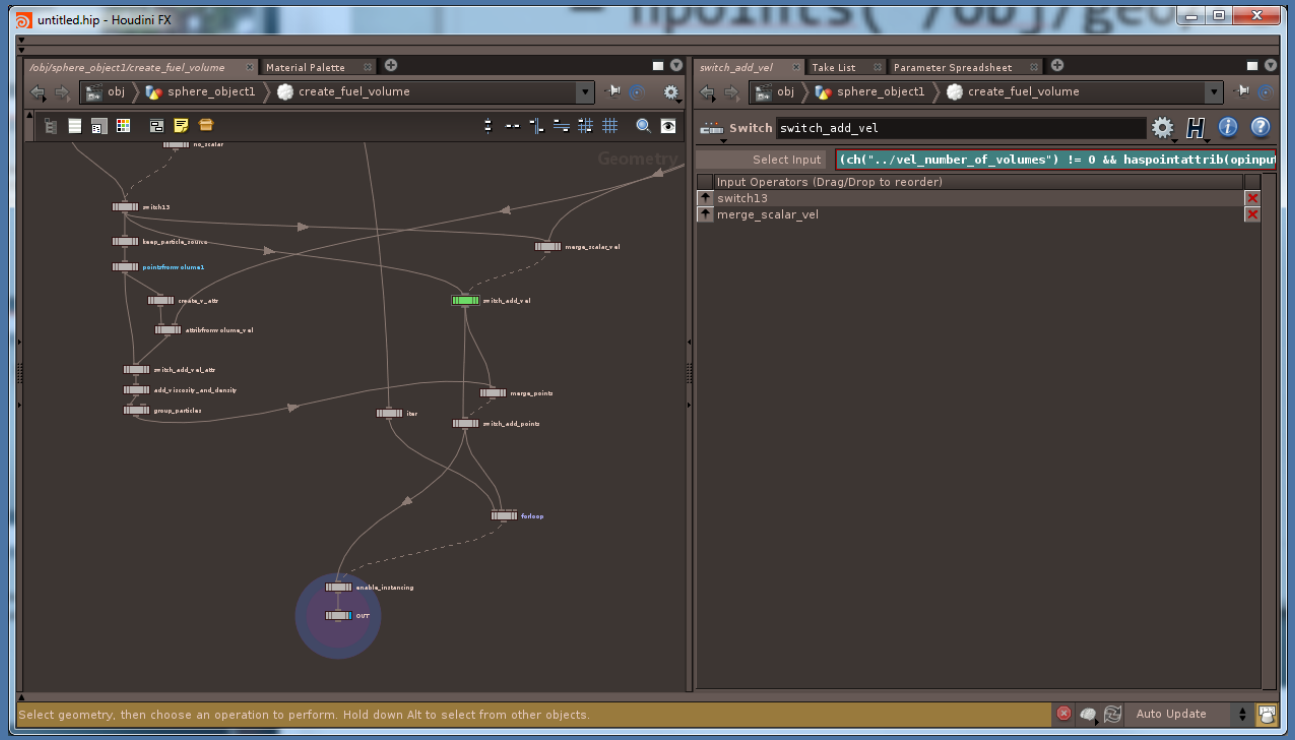
Venerable

- Houdini 1.0 released in 1996
  - Some code survives from original PRISMS
- Core duo released in 2007

Interrelated

- Every part talks to every other part
- This behavior is **not** the exception!

Dependencies are Computed

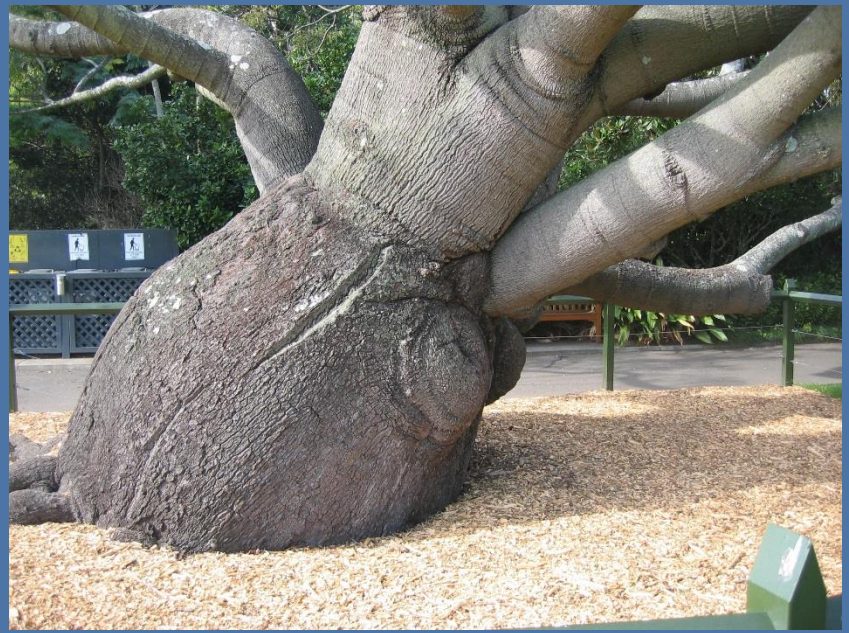


Multithreading in Houdini

Methodology

Incremental Changes

- Continuous integration
- Bend without breaking



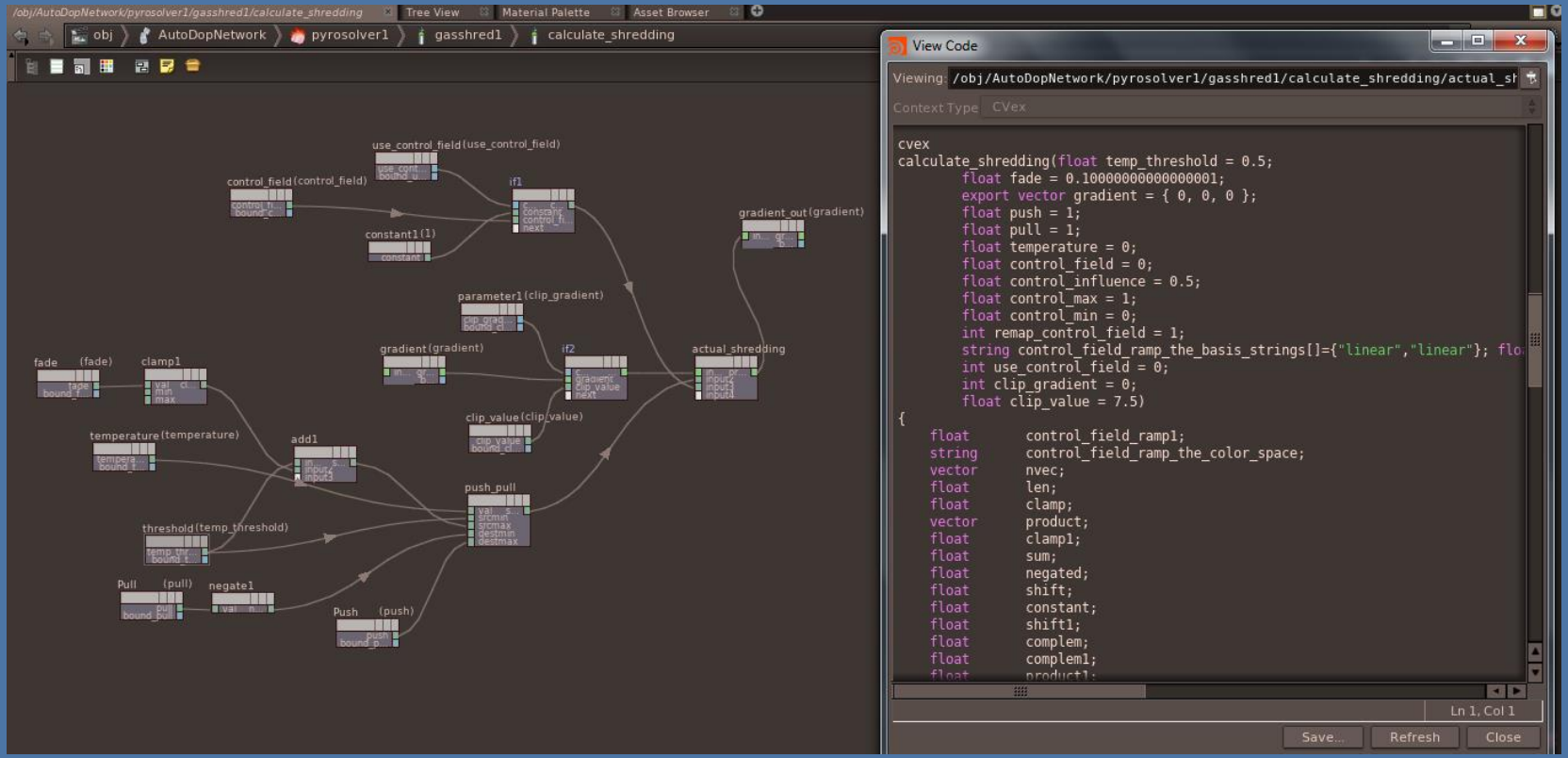
Statics: Not just for physics!

- Grepping for statics
  - `nm libGEO.so | c++filt | grep -i '^[0-9a-F]* [bcdgs]‘`
  - Need additional filters for guards, vtables, etc.
- Intel Compiler warnings
  - `icc –ww1711 –c foo.cc`
  - Warns about writes to statically allocated data
  - Avoids false positives about const-like structures
  - ```
    #define TLA_THREADSAFE_STATIC_WRITE(code) \
    __pragma(warning(disable:1711)); \
    CODE; \
    __pragma(warning(default:1711))
    ```
    ```

Focusing on the Stupidly Parallel

- Make it Easy
  - To Write
  - To Convert
  - To Undo

Stupidly Parallel: VEX



Stupidly Parallel: Task vs Thread

- Task Based
  - tbb::parallel_for
  - Functor runs once per chunk
  - Better load balancing
- Thread Based
  - Thread-pools like UT_ThreadedAlgorithm
  - Functor runs once per thread
  - Less marshalling

Stupidly Parallel: Thread

- For existing code, parameter list is often already minimized
  - The state before the critical for loop is often not!
- Task based functors require marshalling parms and locals
- The inner loop is textually separated from the setup code

```
void foo(parmlist)
{
	// Setup
	float localvars = . . .;

	// execute
	for all elements()
	{
		Code;
	}
}
```

Stupidly Parallel: Task

- UTparallelFor
  - Mandatory grain size
  - Early exit to inline version
- UTparallelForLightItems
- UTparallelForHeavyItems
- UTserialFor

Multithreading in Houdini

Patterns

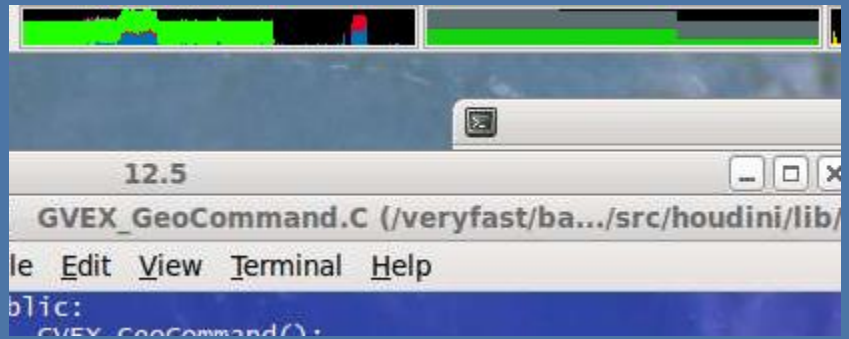
Reentrancy

- Always be reentrant.

```
class foo
{
public:
	void interface1()
	{
		UT_Lock::Scope scope(ourlock);
		// Do computation.
	}
	void interface2()
	{
		UT_Lock::Scope scope(ourlock);
		// Reuse interface1
		interface1()
	}
};
```

Never Lock

- Everything you were taught about clever synchronization is irrelevant
- You do not own the CPU
- You do not even know if there is a CPU



Atomics are Slow

- Treat them as guaranteed uncached memory

```
class FOO_Array
{
public:
	shared_ptr<float  > getData() const
	{ return myData; }
private:
	shared_ptr<float   > myData;
};

float
sum_array(const FOO_Array &foo)
{
	float total = 0;
	for (int i = 0; i < foo.length(); i++)
		total += foo.getData()[i];
}
```

Never Blindly Fork

- Always consider grain size
- Always consider single vs multithreaded

Determinism

- Produce same results regardless of lock resolution.
  - Without this, debugging is a nightmare!
- Produce same results regardless of machine architecture (# threads, etc)
  - Task based is useful
  - We don’t do very well at this

Command Line Control

- -j *ncpu*
- Allows multiple processes to share a machine
- Allows you to easily debug and perf test

Memory Constant in Cores

- How to multithread a random write pattern?
  - Lock writes
  - Find a pattern that is thread safe
  - Copy the destination per thread and merge after
- What if you have a four-socket, ten-core, machine with hyperthreading?

Memory Allocation

- tbb::scalable_malloc fragments
- jemalloc
  - http://www.canonware.com/jemalloc

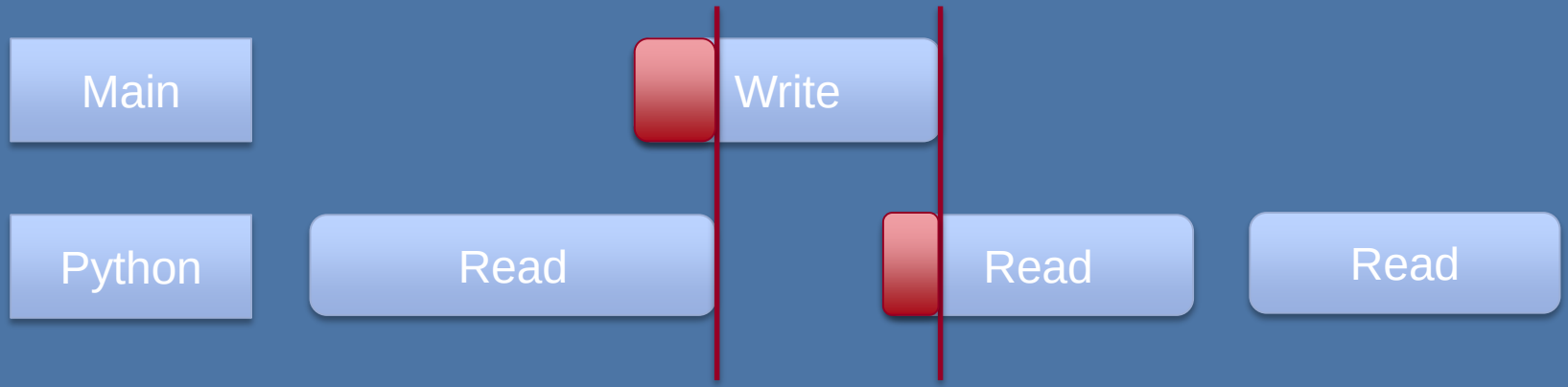
Copy on Write

- Reader/Writer Locks
- Const Correctness
- Importance of Ownership
- Sole Ownership is a Writer Lock
- How this Fails

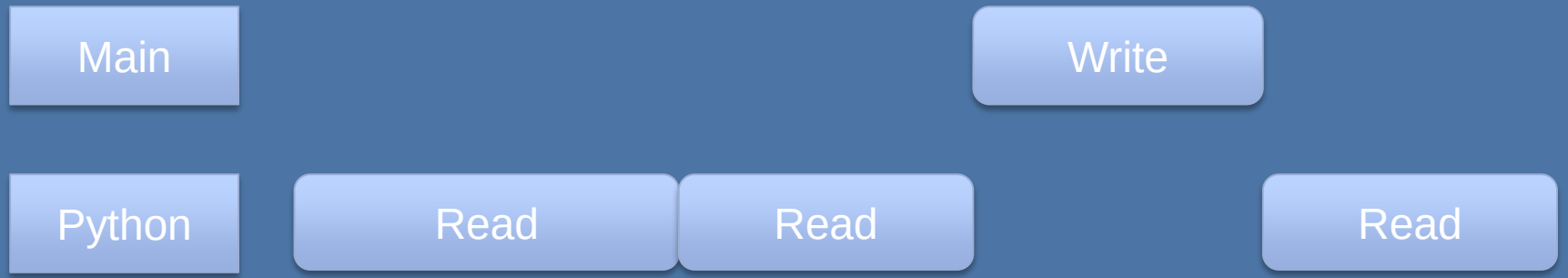
Copy on Write: Reader/Writer

- Natural needs for a piece of geometry
  - Many readers
  - Many writers
- Need to ensure readers are not made inconsistent by writes
- Reader/Writer Lock is a solution
  - But we do not want any locks!

Shared Geometry Cache



Shared Geometry Cache



Barring other synchronization primitives;
this is also a possible execution order.

Shared Geometry Cache

Main

Write Copy

Python

Read

Read Original

Read New

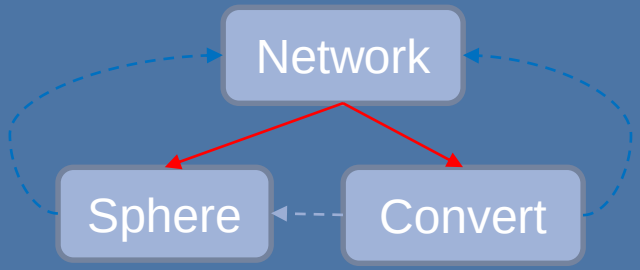
This has the same effect but with overlapping computation.

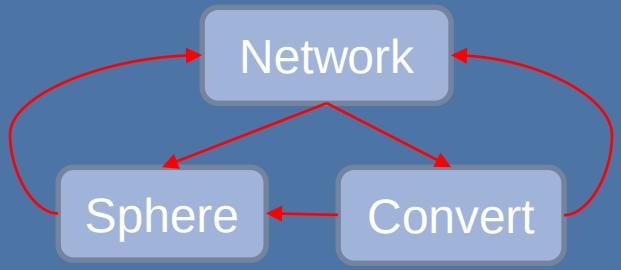
Copy on Write: Const

- Const Correctness
  - Does not provide speed improvements
  - Does create a compiler-enforced contract on the code!
- Easy to find lazy programming
  - const_cast
  - mutable
  - Most importantly, easier to just not make things const!

Copy on Write: Ownership

- Lack of garbage collection is a feature





Copy on Write: Ownership

- shared_ptr should be minimized!

```
class FOO_Array
{
public:
        float *getData() const
        { return myData.get(); }

        shared_ptr<float  > copyData() const
        { return myData; }
private:
        shared_ptr<float   > myData;
};
```

Copy on Write: Sole Ownership

- Two types of readers
  - Readers from foreign threads/algorithms
  - Our own readers
- Preventing external readers lets you reason about your own readers
- Sole ownership is a Writer Lock
  - If no external algorithm has a copy of your data structure, you can reason about a lock-free approach

Copy on Write: Sole Ownership

```
class FOO_Array
{
public:
      const float *readData() const
      { return myData.get(); }

      float *writeData()
      { makeUnique(); return myData.get(); }

      shared_ptr<float  > copyData() const
      { return myData; }

private:
      void makeUnique()
      {
          if (myData.unique()) return;

          shared_ptr<float  > copy(new float*[size];);
          memcpy(copy.get(), myData.get(),
                    sizeof(float)*size);
          myData = copy;
      }


      shared_ptr<float  > myData;
};
```

Copy on Write: Failure

- Excessive sharing

```
void applyPartial(FOO_Array foo, RANGE partialrange)
{
	float *dst = foo.writeData();

	for (i in partialrange)
	{
		dst[i] *= 2;
	}
}

FOO_Array bar;

invoke_parallel(bar, applyPartial);
```

Copy on Write: Failure

- Pointer aliasing

```
void apply(float *dst, const float *src)
{
     for (i = 0; i < size; i++)
     {
          dst[i] = src[i/2];
     }
}

void process(FOO_Array &foo)
{
     const float *src = foo.readData();
     float *dst = foo.writeData();

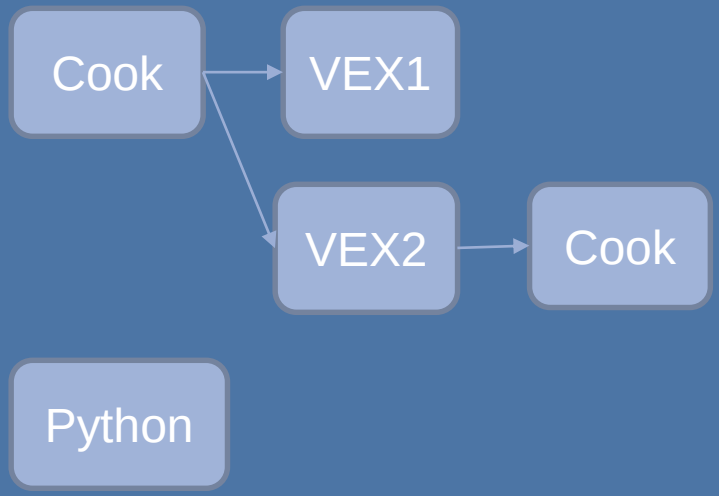
     apply(dst, src);
};
```

Multithreading in Houdini

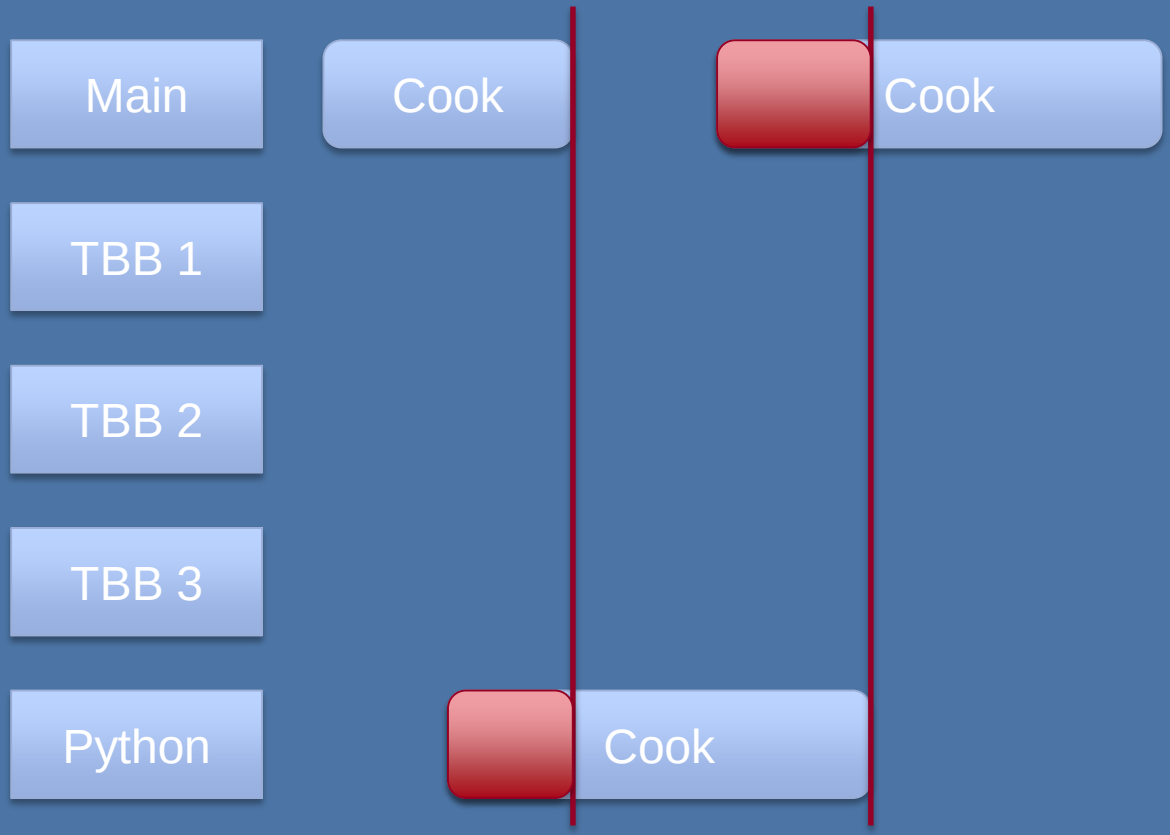
Task Locks

Task Locks

- Not all threads are equal
- A cook lock block Python from reading intermediate state
- VEX tasks spawn to many threads
- VEX can then trigger cooks
- Which needs the cook lock!
- Task locks allow reentrancy to child tasks, even if they are currently on a different thread



Global Cook Lock



Global Cook Lock

Main

Cook

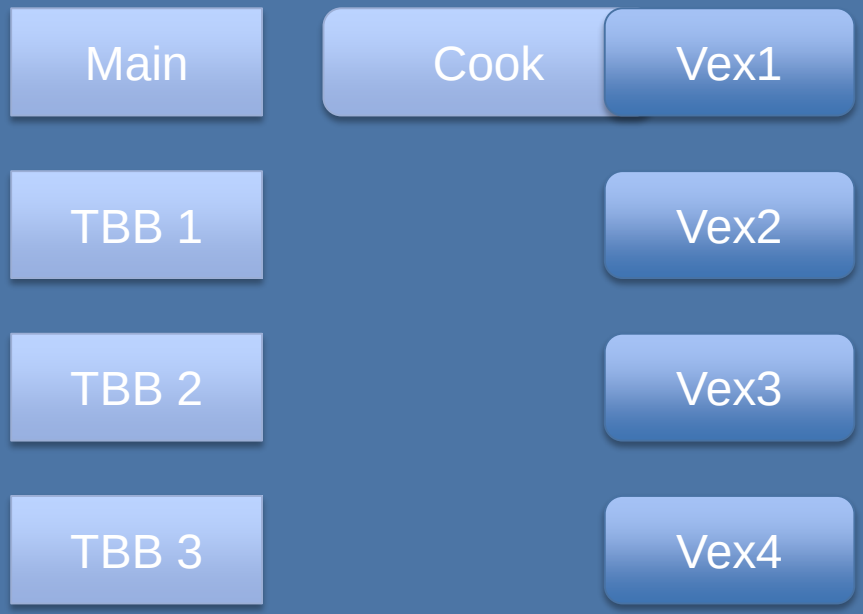
TBB 1

TBB 2

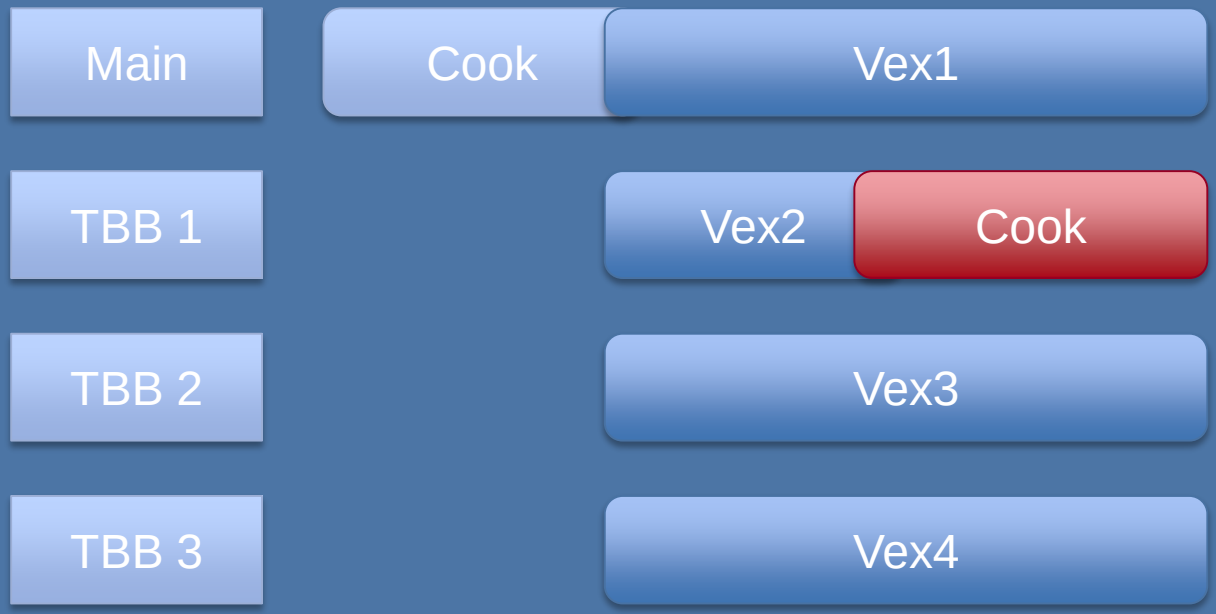
TBB 3

Posted Tasks
Vex1, Vex2, Vex3, Vex4

Global Cook Lock



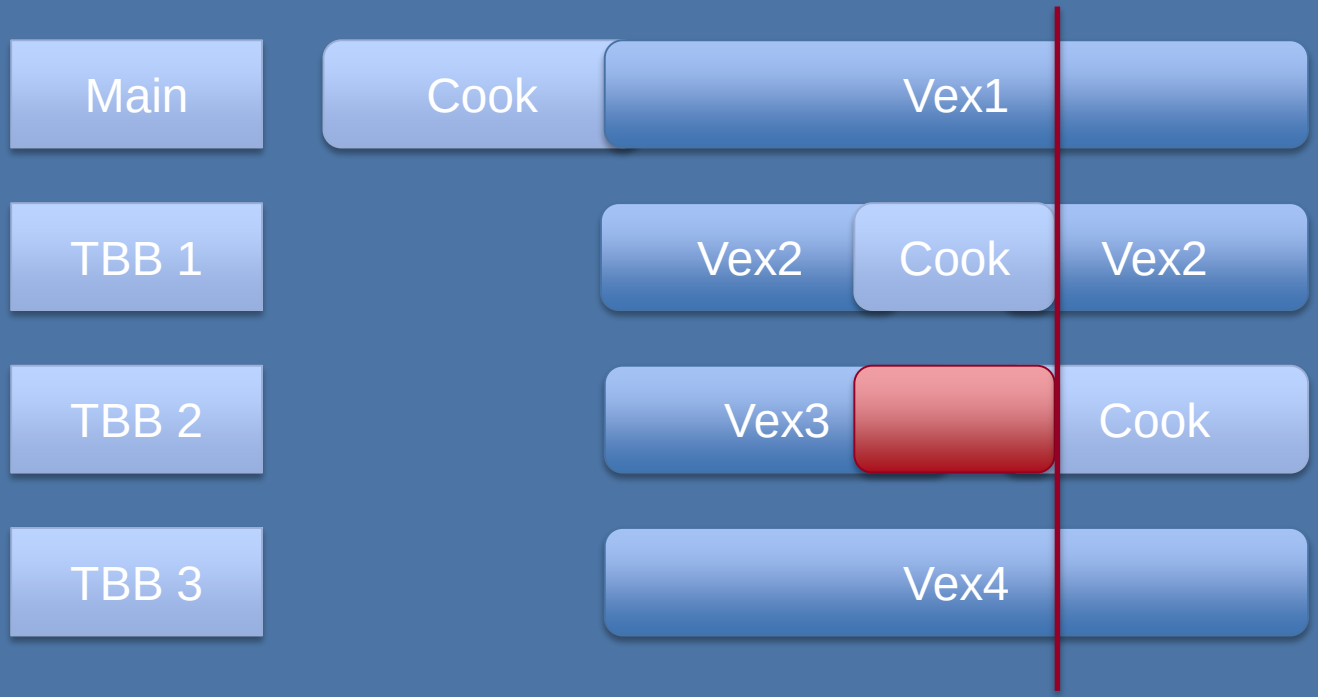
Global Cook Lock



Task Lock: Details

- Task Owner defines a tree of child tasks
- Thread Local stores current thread's Task Owner
- Tasks update their thread's owner on start
- Lock is a condition + mutex which allows a stack of locks provided owner is strictly a child

Task Cook Lock



Composability

Main

Cook

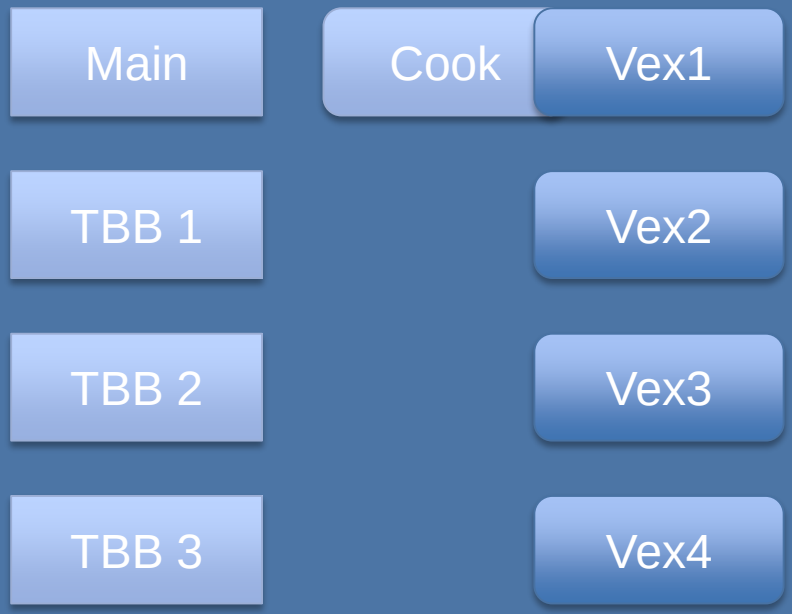
TBB 1

TBB 2

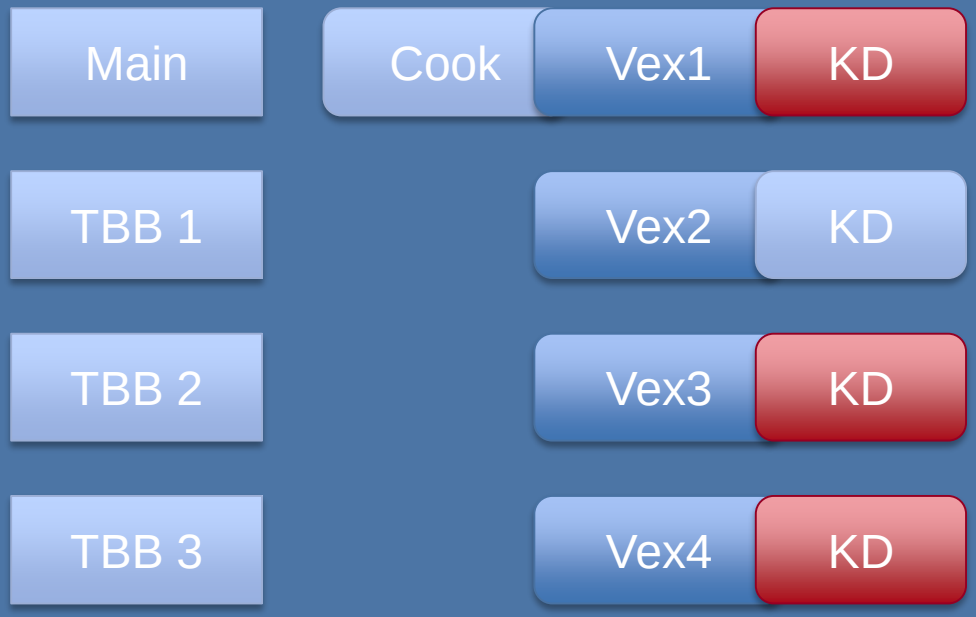
TBB 3

Posted Tasks
Vex1, Vex2, Vex3, Vex4

Composability

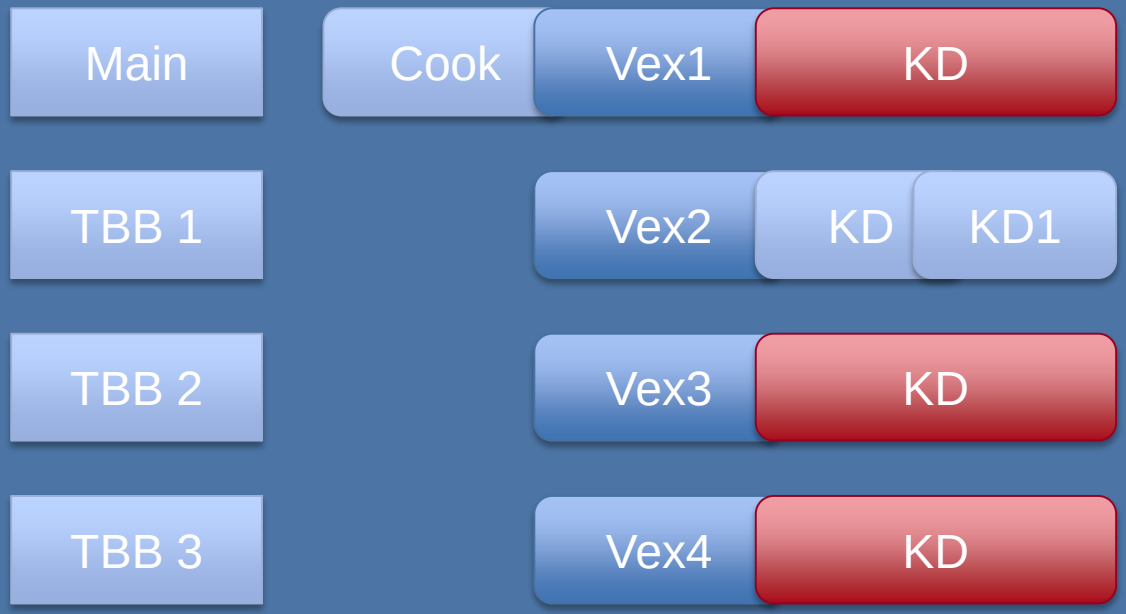


Composability



Posted Tasks
KD1, KD2, KD3, KD4

Composability



Posted Tasks
KD2, KD3, KD4

Composability

How we hope it works…

Deadlock…

Main

Cook

TBB 1

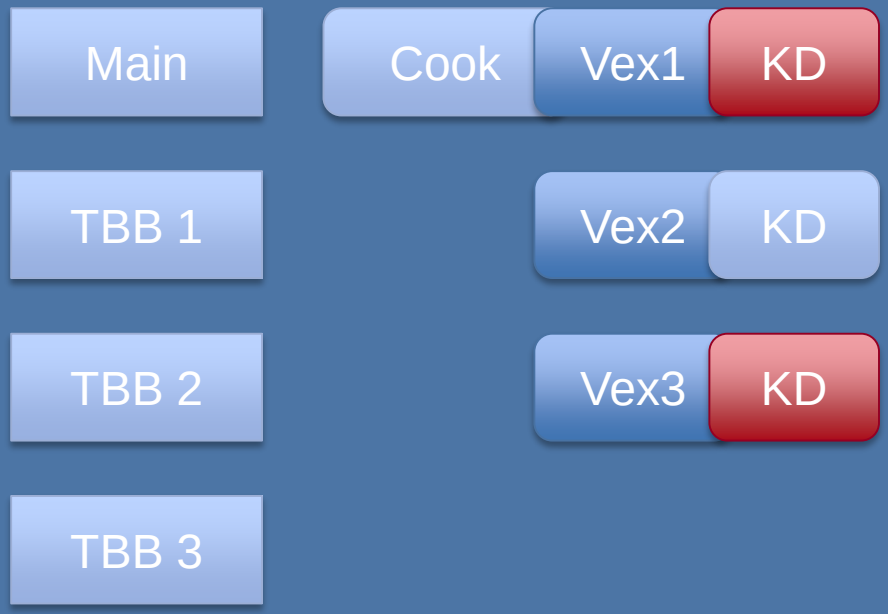
TBB 2

TBB 3

Posted Tasks
Vex1, Vex2, Vex3, Vex4

Deadlock…

Deadlock…



Posted Tasks
Vex4, KD1, KD2, KD3, KD4

Deadlock…

Posted Tasks
Vex4, KD3, KD4

Deadlock…


Main
Cook
Vex1
KD
TBB 1
Vex2
KD
KD1
Vex4
TBB 2
Vex3
KD
TBB 3
KD2
KD3
Posted Tasks
KD4

Deadlock…


Main
Cook
Vex1
KD
TBB 1
Vex2
KD
KD1
Vex4
KD
TBB 2
Vex3
KD
TBB 3
KD2
KD3
KD4

Task Locks

- Task stealing leads to deadlocks
- The stack creates an implicit lock!

Multithreading in Houdini

Task Arenas

Task Arena

- Never call back to the TBB scheduler if a lock is held
- If you must, wrap a Task Arena

Task Arena

- Moved from Preview to mainline in TBB 4.3
- Documentation suggests for controlling thread pools
- Solves the locking problem

Task Arena

- Acquire your lock
- Build just-in-time a tbb::task_arena
- Execute code (task_arena::execute())
  - No external tasks will be assigned to this thread!
- Release your lock

Task Arena

Main

Cook

TBB 1

TBB 2

TBB 3

Posted Tasks
Vex1, Vex2, Vex3, Vex4

Task Arena

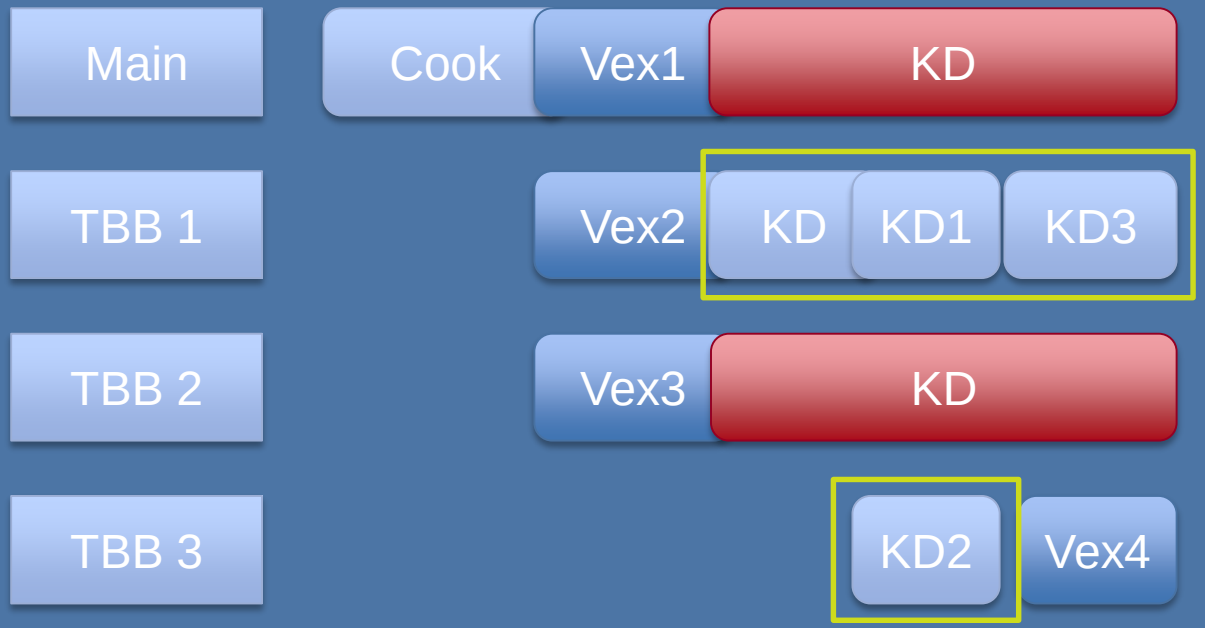
Task Arena

Posted Tasks
Vex4, KD1, KD2, KD3, KD4

Task Arena

Posted Tasks
Vex4, KD3, KD4

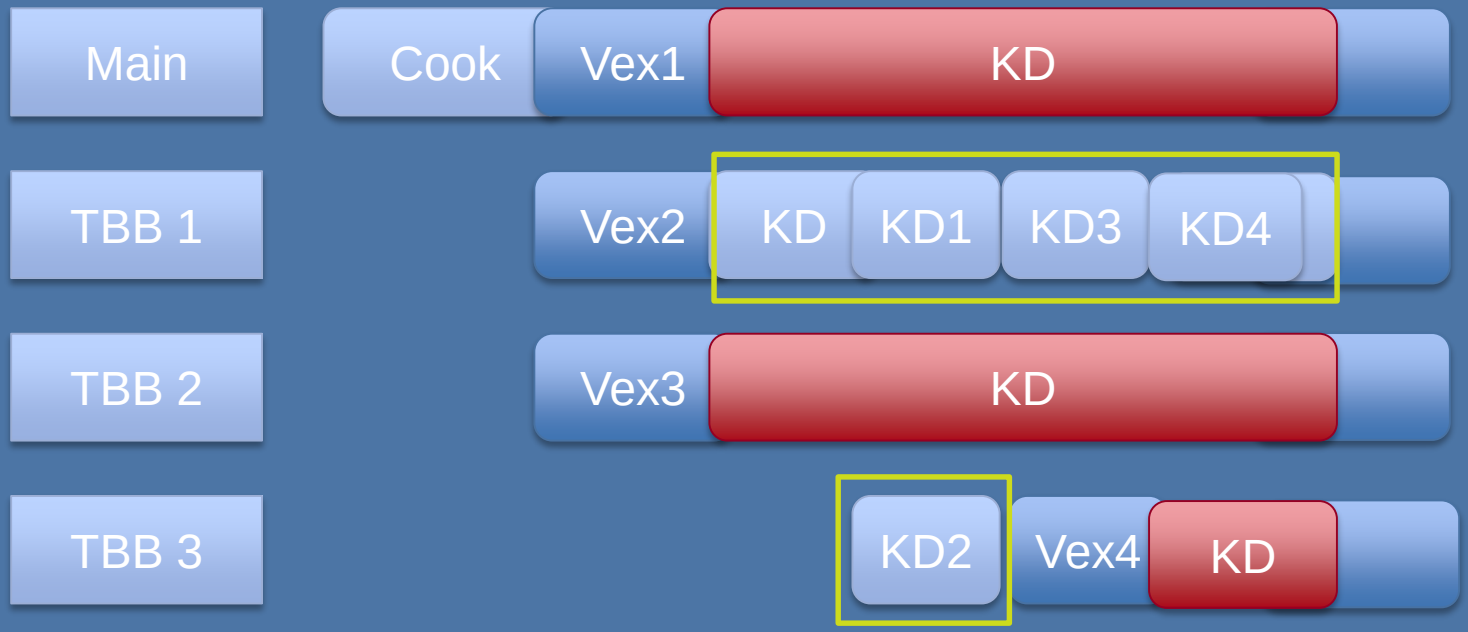
Task Arena



Posted Tasks
KD4

Vex4 CANNOT be assigned to TBB1 until TBB1 leaves the arena!

Task Arena


Main
TBB 1
TBB 2
TBB 3
Cook
Vex1
KD
Vex2
KD
KD1
KD3
KD4
Vex3
KD
KD2
Vex4
KD

Task Arena

- Beware implicit locks from stack unwinding
- Do not re-use
- Catch exceptions

Multithreading in Houdini

Task Exclusive

Task Exclusive

- Many threads want the same cached item
- Task Lock results in single threaded computation of the item.
- Why can’t blocked threads contribute work?

Task Exclusive

- Lock Free Singleton
- Tasks compete to get right to compute resource
- Winning task computes as normal
- Losing tasks add a dependent task and return to scheduler

Task Exclusive

Main

Cook

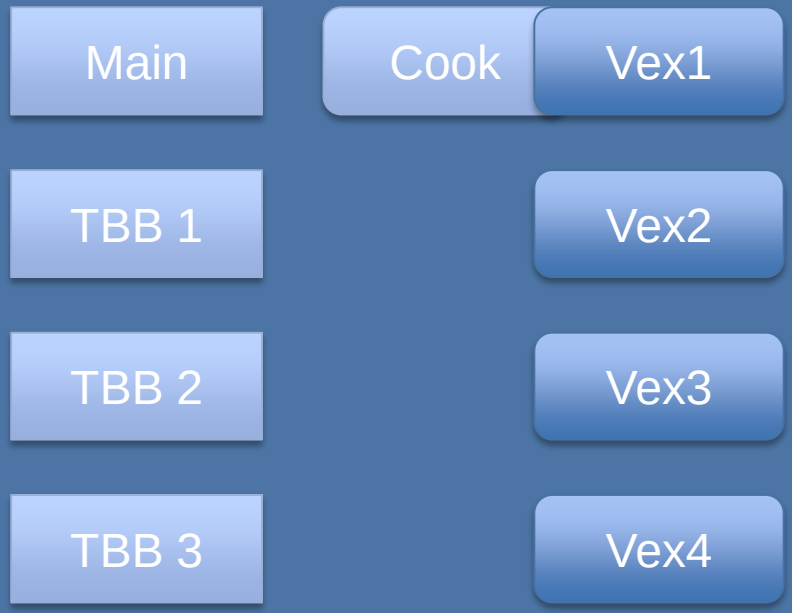
TBB 1

TBB 2

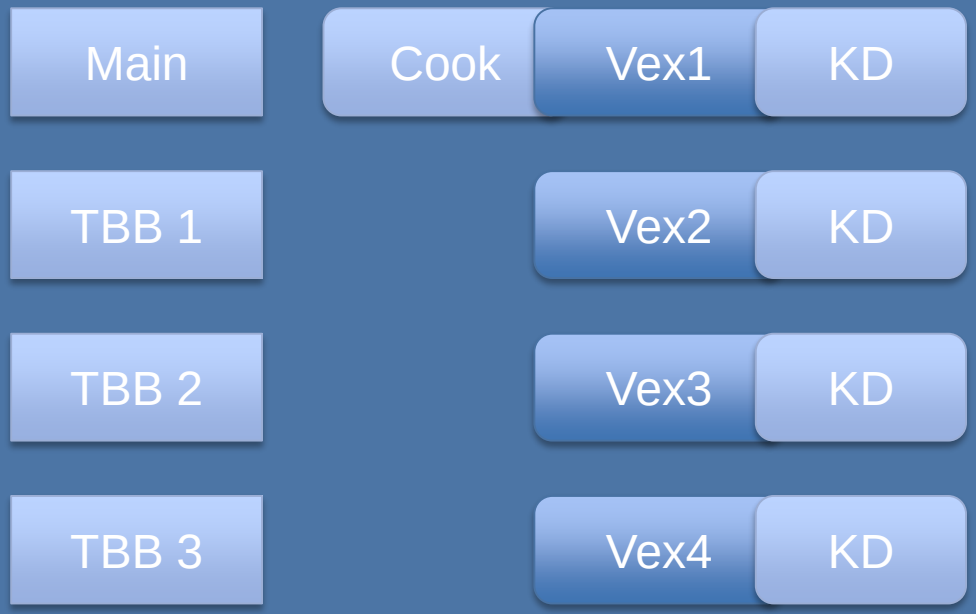
TBB 3

Posted Tasks
Vex1, Vex2, Vex3, Vex4

Task Exclusive



Task Exclusive



Posted Tasks
rKD1, rKD2, rKD3, rKD4

Task Exclusive

Task Exclusive

Posted Tasks
KD1, KD2, KD3, KD4

Waiting on rKD3:
rKD1, rKD2, rKD4

Task Exclusive

Posted Tasks

Waiting on rKD3:
rKD1, rKD2, rKD4

Task Exclusive

Task Exclusive

- Lock Free Singleton
  - But the stack creates an implicit lock!
  - Need a Task Arena to ensure stack can empty back to the recycled task.

Task Exclusive

Stack Locks

```
(gdb) info threads
  Id Target Id Frame
  17 Thread 0x7f9c2c603700 (LWP 22414) "houdini" 0x00007f9c598a3de3 in select
    () at ../sysdeps/unix/syscall-template.S:81
  16 Thread 0x7f9c21478700 (LWP 22482) "houdini" pthread_cond_wait@@GLIBC_2.3.2 () at ../n
  15 Thread 0x7f9c18435700 (LWP 22485) "houdini" syscall ()
    at ../sysdeps/unix/sysv/linux/x86_64/syscall.S:38
  14 Thread 0x7f9c18034700 (LWP 22486) "houdini" syscall ()
    at ../sysdeps/unix/sysv/linux/x86_64/syscall.S:38
* 13 Thread 0x7f9c17c33700 (LWP 22487) "houdini" 0x00007f9c5987ccf7 in sched_yield () at ../sy
  12 Thread 0x7f9c17832700 (LWP 22488) "houdini" syscall ()
    at ../sysdeps/unix/sysv/linux/x86_64/syscall.S:38
  11 Thread 0x7f9c17431700 (LWP 22489) "houdini" syscall ()
    at ../sysdeps/unix/sysv/linux/x86_64/syscall.S:38
  10 Thread 0x7f9c15fda700 (LWP 22490) "houdini" syscall ()
    at ../sysdeps/unix/sysv/linux/x86_64/syscall.S:38
  9 Thread 0x7f9c153d8700 (LWP 22491) "houdini" syscall ()
    at ../sysdeps/unix/sysv/linux/x86_64/syscall.S:38
  8 Thread 0x7f9c157d9700 (LWP 22493) "houdini" syscall ()
    at ../sysdeps/unix/sysv/linux/x86_64/syscall.S:38
  7 Thread 0x7f9c14cd7700 (LWP 22492) "houdini" syscall ()
    at ../sysdeps/unix/sysv/linux/x86_64/syscall.S:38
  6 Thread 0x7f9c148d6700 (LWP 22494) "houdini" syscall ()
    at ../sysdeps/unix/sysv/linux/x86_64/syscall.S:38
  5 Thread 0x7f9c144d5700 (LWP 22495) "houdini" __TBB_machine_pause (
    delay=80) at ../../include/tbb/machine/gcc_ia32_common.h:55
  4 Thread 0x7f9be7fff700 (LWP 22496) "houdini" 0x00007f9c598a3de3 in select
    () at ../sysdeps/unix/syscall-template.S:81
  3 Thread 0x7f9be6ffe700 (LWP 22500) "houdini" 0x00007f9c598a3de3 in select
    () at ../sysdeps/unix/syscall-template.S:81
  2 Thread 0x7f9be5d52700 (LWP 22661) "houdini" 0x00007f9c598a3de3 in select
    () at ../sysdeps/unix/syscall-template.S:81
  1 Thread 0x7f9c5bf0a840 (LWP 22411) "houdini" 0x00007f9c5987ccf7 in sched_yield () at ../sysd
```

Stack Locks


#0 0x00007f9c5987ccf7 in sched_yield ()
at ../sysdeps/unix/syscall-template.S:81
#1 0x00007f9c4cedb3f5 in tbb::internal::custom_scheduler<tbb::internal::IntelSchedulerTraits>::receive_or_steal_task (this=0x7f9c349a3d00,
completion_ref_count=@0x7f9c3498ba28: 2, return_if_no_work=false)
at ../../src/tbb/custom_scheduler.h:291
#2 0x00007f9c4cedba5e in tbb::internal::custom_scheduler<tbb::internal::IntelSchedulerTraits>::local_wait_for_all (this=0x7f9c349a3d00, parent=...,
child=<optimized out>) at ../../src/tbb/custom_scheduler.h:582
#3 0x00007f9c4ceda2d7 in tbb::internal::generic_scheduler::local_spawn_root_and_wait (this=0x7f9c349a3d00, first=..., next=@0x7f9c34089638: 0x7f9c3498ba40
at ../../src/tbb/scheduler.cpp:661
#4 0x00007f9c437c8984 in execute (func=..., this=0x7f9b3cda88e0)
at /superfast/baselines/dev_blue/hfs/custom/houdini/include/UT/UT_TaskExclusive.h:91
#5 deferredBuild (geo=0x7f9b3cda88c0, this=0x7f9b3cda88e0)
at GVEX_GeoCache.C:228
#6 buildPointTree (this=0x7f9b3cda88c0) at GVEX_GeoCache.C:262
#7 GVEX_GeoCache::getPointTree (state=..., file=0x191d2328 "opinput:0",
group=<optimized out>, attrib=0xb17ad58 "P") at GVEX_GeoCache.C:2881
#8 0x00007f9c43775795 in vexPCFindAISSVFI<0, 0> (state=..., proc=...)
at GVEX_Functions.C:1065
#9 0x00007f9c493e9da4 in executeInstruction (proc=..., state=...,
word=<optimized out>) at VEX_Instance.C:142
#10 executeStrInstruction (state=..., proc=..., off=<optimized out>)
at VEX_Instance.C:204
#11 0x00007f9c493ee9e0 in executeStrInstruction (proc=..., state=...)
at VEX_Instance.C:210
#12 VEX_Instance::runInstanceImpl<false, false> (this=0x7f9a7461a100,
proc=..., data=...) at VEX_Instance.C:340
#13 0x00007f9c42dfa608 in runInstance (data=..., proc=..., this=0x7f9a7461a100)
at /superfast/baselines/dev_blue/hfs/custom/houdini/include/VEX/VEX_Instance.h:276
#14 CVEX_Context::run (this=0x7ffff81e1100, uproc=1024,
interruptable=<optimized out>, cdata=0x7ffff81e0fe8) at CVEX_Context.C:1030
#15 0x00007f9c3838413d in GAS_GeometryVex::processVexBlock (this=0x4,
gdp=<optimized out>, data=..., context=..., argc=16, argv=0x7ffff81e143a0,
geomap=..., num=1024, elementoffsets=0x7f9b2ba49350, triviallist=true,
info=...) at GAS_GeometryVex.C:3517
#16 0x00007f9c3838c32c in GAS_GeometryVex::processVexGeometryPartial (
this=0x7f9b8dd49320, parms=..., info=...) at GAS_GeometryVex.C:1876
#17 0x00007f9c38399129 in GAS_GeometryVex::processVexGeometryInvokePartial (
this=<optimized out>, parms=..., info=...) at GAS_GeometryVex.h:147
#18 0x00007f9c5a4e9d48 in operator() (parm1=..., this=0x7f9c3498a058)
at UT_Functor.h:434
#19 operator() (range=..., this=0x7f9c3498a058) at UT_ThreadedAlgorithm.C:171
#20 run_body (r=..., this=0x7f9c3498a040)
at /superfast/baselines/dev_blue/hfs/custom/include/tbb/parallel_for.h:110
#21 tbb::interface6::internal::simple_partition_type::execute<tbb::interface6::internal::start_for<tbb::blocked_range<int>, ut_ApplyFunctor, tbb::simple_partitioner c
range=...)
at /superfast/baselines/dev_blue/hfs/custom/include/tbb/partitioner.h:428
#22 0x00007f9c5a4ea1e4 in tbb::interface6::internal::start_for<tbb::blocked_range<int>, ut_ApplyFunctor, tbb::simple_partitioner const>::execute (
this=<optimized out>)
at /superfast/baselines/dev_blue/hfs/custom/include/tbb/parallel_for.h:116
#23 0x00007f9c4cedba7a in tbb::internal::custom_scheduler<tbb::internal::IntelSchedulerTraits>::local_wait_for_all (this=0x7f9c349a3d00, parent=...,
child=<optimized out>) at ../../src/tbb/custom_scheduler.h:472
#24 0x00007f9c4ceda2d7 in tbb::internal::generic_scheduler::local_spawn_root_and_wait (this=0x7f9c349a3d00, first=..., next=@0x7f9c3498b938: 0x7f9c34989e40
at ../../src/tbb/scheduler.cpp:661
#25 0x00007f9c437c8984 in execute (func=..., this=0x7f9b3cda88e0)
at /superfast/baselines/dev_blue/hfs/custom/houdini/include/UT/UT_TaskExclusive.h:91
#26 deferredBuild (geo=0x7f9b3cda88c0, this=0x7f9b3cda88e0)
at GVEX_GeoCache.C:228
#27 buildPointTree (this=0x7f9b3cda88c0) at GVEX_GeoCache.C:262
#28 GVEX_GeoCache::getPointTree (state=..., file=0x191d2328 "opinput:0",
group=<optimized out>, attrib=0xb17ad58 "P") at GVEX_GeoCache.C:2881
#29 0x00007f9c43775795 in vexPCFindAISSVFI<0, 0> (state=..., proc=...)
at GVEX_Functions.C:1065
#30 0x00007f9c493e9da4 in executeInstruction (proc=..., state=...,
word=<optimized out>) at VEX_Instance.C:142
#31 executeStrInstruction (state=..., proc=..., off=<optimized out>)
at VEX_Instance.C:204
#32 0x00007f9c493ee9e0 in executeStrInstruction (proc=..., state=...)
at VEX_Instance.C:210
#33 VEX_Instance::runInstanceImpl<false, false> (this=0xb1450d0, proc=...,
data=...) at VEX_Instance.C:340
#34 0x00007f9c42dfa608 in runInstance (data=..., proc=..., this=0xb1450d0)
at /superfast/baselines/dev_blue/hfs/custom/houdini/include/VEX/VEX_Instance.h:276
#35 CVEX_Context::run (this=0x7ffff81e2100, uproc=1024,
interruptable=<optimized out>, cdata=0x7ffff81e1fe8) at CVEX_Context.C:1030
#36 0x00007f9c3838413d in GAS_GeometryVex::processVexBlock (this=0x4,

Stack Locks


#0 0x00007f9c5987ccf7 in sched_yield ()
at ../sysdeps/unix/syscall-template.S:81
#1 0x00007f9c4cedb1f5 in tbb::internal::custom_scheduler<tbb::internal::IntelSchedulerTraits>::receive_or_steal_task (this=0x7f9c349a3d00,
completion_ref_count=@0x7f9c3498ba28: 2, return_if_no_work=false)
at ../../src/tbb/custom_scheduler.h:291
#2 0x00007f9c4cedba5e in tbb::internal::custom_scheduler<tbb::internal::IntelSchedulerTraits>::local_wait_for_all (this=0x7f9c349a3d00, parent=...,
child=<optimized out>) at ../../src/tbb/custom_scheduler.h:582
#3 0x00007f9c4ceda2d7 in tbb::internal::generic_scheduler::local_spawn_root_and_wait (this=0x7f9c349a3d00, first=..., next=@0x7f9c34989638: 0x7f9c3498ba40)
at ../../src/tbb/scheduler.cpp:661
#4 0x00007f9c437c8984 in execute (func=..., this=0x7f9b3cda88c0)
at /superfast/baselines/dev_blue/hfs/custom/houdini/include/UT/UT_TaskExclusive.h:91
#5 doDeferredBuild (geo=0x7f9b3cda88c0, this=0x7f9b3cda88e0)
at GVEX_GeoCache.C:228
#6 buildPointTree (this=0x7f9b3cda88c0) at GVEX_GeoCache.C:262
#7 GVEX_GeoCache::getPointTree (state=..., file=0x191d2328 "opinput:0",
group=<optimized out>, attrib=0xb17ad58 "P") at GVEX_GeoCache.C:2881
#8 0x00007f9c43775795 in vexPCFindAISSVFI<0, 0> (state=..., proc=...)
at GVEX_Functions.C:1065
#9 0x00007f9c493e9da4 in executeInstruction (proc=..., state=...,
word=<optimized out>) at VEX_Instance.C:142
#10 executeStrInstruction (state=..., proc=..., off=<optimized out>)
at VEX_Instance.C:204
#11 0x00007f9c493ee9e0 in executeStrInstruction (proc=..., state=...)
at VEX_Instance.C:210
#12 VEX_Instance::runInstanceImpl<false, false> (this=0x7f9a7461a100,
proc=..., data=...) at VEX_Instance.C:340
#13 0x00007f9c42dfa608 in runInstance (data=..., proc=..., this=0x7f9a7461a100)
at /superfast/baselines/dev_blue/hfs/custom/houdini/include/VEX/VEX_Instance.h:276
#14 CVEX_Context::run (this=0x7ffff81e1100, nproc=1024,
interruptable=<optimized out>, cdata=0x7ffff81e0fe8) at CVEX_Context.C:1030
#15 0x00007f9c3838413d in GAS_GeometryVex::processVexBlock (this=0x4,
gdp=<optimized out>, data=..., context=..., argc=16, argv=0x7ffff81e43a0,
geomap=..., num=1024, elementoffsets=0x7f9b2ba49350, triviallist=true,
info=...) at GAS_GeometryVex.C:3517
#16 0x00007f9c3838c32c in GAS_GeometryVex::processVexGeometryPartial (
this=0x7f9b8dd49420, parms=..., info=...) at GAS_GeometryVex.C:1876
#17 0x00007f9c38399129 in GAS_GeometryVex::processVexGeometryInvokePartial (
this=<optimized out>, parms=..., info=...) at GAS_GeometryVex.h:147
#18 0x00007f9c5a4e9d48 in operator() (parm1=..., this=0x7f9c3498a058)
at UT_Functor.h:434
#19 operator() (range=..., this=0x7f9c3498a058) at UT_ThreadedAlgorithm.C:171
#20 run_body (r=..., this=0x7f9c3498a040)
at /superfast/baselines/dev_blue/hfs/custom/include/tbb/parallel_for.h:110
#21 tbb::interface6::internal::simple_partition_type::execute<tbb::interface6::internal::start_for<tbb::blocked_range<int>, ut_ApplyFunctor, tbb::simple_partitioner c
range=...)
at /superfast/baselines/dev_blue/hfs/custom/include/tbb/partitioner.h:428
#22 0x00007f9c5a4ea1e4 in tbb::interface6::internal::start_for<tbb::blocked_range<int>, ut_ApplyFunctor, tbb::simple_partitioner const>::execute (
this=<optimized out>)
at /superfast/baselines/dev_blue/hfs/custom/include/tbb/parallel_for.h:116
#23 0x00007f9c4cedba7a in tbb::internal::custom_scheduler<tbb::internal::IntelSchedulerTraits>::local_wait_for_all (this=0x7f9c349a3d00, parent=...,
child=<optimized out>) at ../../src/tbb/custom_scheduler.h:472
#24 0x00007f9c4ceda2d7 in tbb::internal::generic_scheduler::local_spawn_root_and_wait (this=0x7f9c349a3d00, first=..., next=@0x7f9c3498b938: 0x7f9c34989e40)
at ../../src/tbb/scheduler.cpp:661
#25 0x00007f9c437c8984 in execute (func=..., this=0x7f9b3cda88e0)
at /superfast/baselines/dev_blue/hfs/custom/houdini/include/UT/UT_TaskExclusive.h:91
#26 doDeferredBuild (geo=0x7f9b3cda88c0, this=0x7f9b3cda88e0)
at GVEX_GeoCache.C:228
#27 buildPointTree (this=0x7f9b3cda88c0) at GVEX_GeoCache.C:262
#28 GVEX_GeoCache::getPointTree (state=..., file=0x191d2328 "opinput:0",
group=<optimized out>, attrib=0xb17ad58 "P") at GVEX_GeoCache.C:2881
#29 0x00007f9c43775795 in vexPCFindAISSVFI<0, 0> (state=..., proc=...)
at GVEX_Functions.C:1065
#30 0x00007f9c493e9da4 in executeInstruction (proc=..., state=...,
word=<optimized out>) at VEX_Instance.C:142
#31 executeStrInstruction (state=..., proc=..., off=<optimized out>)
at VEX_Instance.C:204
#32 0x00007f9c493ee9e0 in executeStrInstruction (proc=..., state=...)
at VEX_Instance.C:210
#33 VEX_Instance::runInstanceImpl<false, false> (this=0xb1450d0, proc=...,
data=...) at VEX_Instance.C:340
#34 0x00007f9c42dfa608 in runInstance (data=..., proc=..., this=0xb1450d0)
at /superfast/baselines/dev_blue/hfs/custom/houdini/include/VEX/VEX_Instance.h:276
#35 CVEX_Context::run (this=0x7ffff81e2100, nproc=1024,
interruptable=<optimized out>, cdata=0x7ffff81e1fe8) at CVEX_Context.C:1030
#36 0x00007f9c3838413d in GAS_GeometryVex::processVexBlock (this=0x4,

Multithreading in Houdini

Conclusion

Final Thoughts

- Make it easy to parallelize
- Make it easy to switch to serial
- Use jemalloc
- You do not need to rewrite

Thank you