Threading Performance Measurements with VTune

Christopher Jones *FNAL*

What is VTune?

VTune is an Intel performance measurement system

Can measure

- Concurrency efficiency
- Cross threads locks and waits
- CPU hardware counters
- GPU performance

Can do measurements from the GUI or command line tool

- Data generated by command line tool can be read into the GUI for analysis

OpenLab has a cross CERN license

- https://twiki.cern.ch/twiki/bin/view/Openlab/IntelTools#Intel_Parallel_Studio_XE_201_AN1

Using VTune with cmsRun

VTune did not work 'out of the box' on cmsRun

Steps needed to work

Had to use most recent VTune beta
- older releases all had assertion failures
- /afs/cern.ch/sw/IntelSoftware/linux/x86_64/xe2016/vtune_amplifier_xe_2016.1.0.424694

Tell VTune to ignore processes started by ROOT
- -strategy=ld-linux.so.2:nt:nt,ld-linux-x86-64.so.2:nt:nt
- VTune is unable to instrument these processes

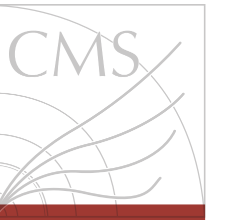
Had to enable signal 38

Problem seen

GUI sometimes only shows measurements from some of the threads

Do not know if failure was during data collection or in viewing

Measurement



Replicated Tier 0 job

cmsRun configuration changes

Max events of 992
Number of threads 8

VTune configuration

concurrency measurement
skip first 5 minutes of job
sample every 100us
unlimited data collection size

VTune GUI

<no current project> - Intel VTune Amplifier (on cmsdev02.cern.ch)

Welcome | r003cc

Concurrency Hotspots by CPU Usage viewpoint (change) — Intel VTune Amplifier XE 2016

Collection Log | Analysis Target | Analysis Type | Summary | Bottom-up | Caller/Callee | Top-down Tree | Platform

Grouping: Function / Call Stack

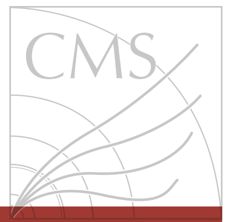
Function / Call Stack	CPU Time: Effective Time by Utilization	CPU Time: Spin Time	CPU Time: Ove.. Time	Wait Time by Utilization
edm::one::OutputModuleBase::doEvent				10569.771s
TObjArray::Delete	2125.000s	0s	0s	6720.931s
full_read.constprop.172				4494.055s
XrdSysTimer::Wait	0.100s	0s	0s	4493.590s
XrdSys::IOEvents::PollE::Begin	0.100s	0s	0s	4492.156s
TClingCallFunc::IFacePtr	1229.920s	0s	0s	3169.468s
TClass::GetCheckSum	973.722s	0s	0s	2351.981s
TClass::GetClass	420.100s	0s	0s	1285.435s
TClass::GetBaseClassOffset	405.951s	0s	0s	1131.577s
Selected 1 row(s):				10569.771s

Data Of Interest (CPU Metrics) — Viewing 0 of 0 selected stack(s) — No stack information

Threads: TBB Worker ... (×7), cmsRunGlib... — Legend: Running, Waits, CPU Time, Spin and ..., CPU Sample, Transitions, CPU Usage, Thread Concurre...

Filter: 100.0% is shown — Process: [100.0%] cmsRunGlibC — Thread: Any Thread — Module: Any Module — Utilization: Any Utilization

VTune GUI



Modes

Summary: high level view of results

Bottom-up: accumulated time in function for all stack traces

Caller/Callee: show time in call stack for selected function

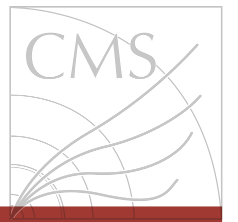
Top-down: tree view of time spent in stack traces

VTune GUI

Function / Call Stack

Function with the time measurements
Triangle reveals call stacks leading to the function

VTune GUI



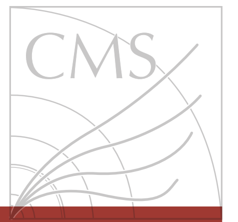
Time

CPU Time: CPU time spent in the function

Wait Time: Time waiting in function summed across threads

Can sort on any of the columns

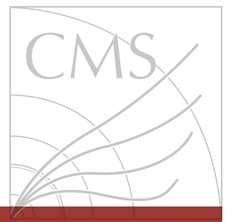
VTune GUI



Thread View

Utilization of each thread over time
Can filter based on time
Can select waits and see function selected above

Initial Measurement



<no current project> - Intel VTune Amplifier (on cmsdev02.cern.ch)

Welcome | r003cc

Concurrency Hotspots by CPU Usage viewpoint (change)

Intel VTune Amplifier XE 2016

Collection Log | Analysis Target | Analysis Type | Summary | Bottom-up | Caller/Callee | Top-down Tree | Platform

Grouping: Function / Call Stack

Function / Call Stack	CPU Time: Effective Time by Utilization (Idle, Poor, Ok, Ideal, Over)	CPU Time: Spin Time	CPU Time: Ove.. Time	Wait Time by Utilization (Idle, Poor, Ok, Ideal, Over)
edm::one::OutputModuleBase::doEvent				10569.771s
TObjArray::Delete	2125.000s	0s	0s	6720.931s
full_read.constprop.172				4494.055s
XrdSysTimer::Wait	0.100s	0s	0s	4493.590s
XrdSys::IOEvents::PollE::Begin	0.100s	0s	0s	4492.156s
TClingCallFunc::IFacePtr	1229.920s	0s	0s	3169.468s
TClass::GetCheckSum	973.722s	0s	0s	2351.981s
TClass::GetClass	420.100s	0s	0s	1285.435s
TClass::GetBaseClassOffset	405.951s	0s	0s	1131.577s
Selected 1 row(s):				10569.771s

Data Of Interest (CPU Metrics)

Viewing 0 of 0 selected stack(s)

No stack information

500s 1000s 1500s 2000s 2500s 3000s 3500s 4000s 4500s 5000s

Thread: TBB Worker ... | TBB Worker ... | TBB Worker ... | TBB Worker ... | TBB Worker ... | TBB Worker ... | TBB Worker ... | cmsRunGlib...

paused

Thread: Running, Waits, CPU Time, Spin and ..., CPU Sample, Transitions, CPU Usage, Thread Concurre...

Filter: 100.0% is shown | Process: [100.0%] cmsRunGlibC | Thread: Any Thread | Module: Any Module | Utilization: Any Utilization

Initial Measurement

<no current project> - Intel VTune Amplifier (on cmsdev02.cern.ch)

Welcome | r003cc

Concurrency Hotspots by CPU Usage viewpoint (change)

Intel VTune Amplifier XE 2016

Collection Log | Analysis Target | Analysis Type | Summary | Bottom-up | Caller/Callee | Top-down Tree | Platform

Grouping: Function / Call Stack

Poor Thread Utilization

Only short times of good CPU Time

Lots of waiting

Function / Call Stack	CPU Time			Wait Time by Utilization
edm::one::OutputModuleBase::doEvent				
TObjArray::Delete				
full_read.constprop.172				
XrdSysTimer::Wait				
XrdSys::IOEvents::PollE::Begin				
TClingCallFunc::IFacePtr	1229.920s	0s	0s	3169.468s
TClass::GetCheckSum	973.722s	0s	0s	2351.981s
TClass::GetClass	420.100s	0s	0s	1285.435s
TClass::GetBaseClassOffset	405.951s	0s	0s	1131.577s
Selected 1 row(s):				10569.771s

Data Of Interest (CPU Metrics)

Viewing 0 of 0 selected stack(s)

No stack information

Threads: TBB Worker ... (×7), cmsRunGlib...

Timeline: 500s 1000s 1500s 2000s 2500s 3000s 3500s 4000s 4500s 5000s

Thread: Running, Waits, CPU Time, Spin and ..., CPU Sample, Transitions, CPU Usage, Thread Concurre...

Filter: 100.0% is shown | Process: [100.0%] cmsRunGlibC | Thread: Any Thread | Module: Any Module | Utilization: Any Utilization

Initial Measurement

<no current project> - Intel VTune Amplifier (on cmsdev02.cern.ch)

Welcome | r003cc

Concurrency Hotspots by CPU Usage viewpoint (change) — Intel VTune Amplifier XE 2016

Collection Log | Analysis Target | Analysis Type | Summary | Bottom-up | Caller/Callee | Top-down Tree | Platform

Grouping: Function / Call Stack

Function / Call Stack	CPU Time: Effective Time by Utilization	Spin Time	Ove.. Time	Wait Time by Utilization
edm::one::OutputModuleBase::doEvent				10569.771s
TObjArray::Delete	2125.000s	0s	0s	6720.931s
full_read.constprop.172				4494.055s
XrdSysTimer::Wait	0.100s	0s	0s	4493.590s
XrdSys::IOEvents::PollE::Begin	0.100s	0s	0s	4492.156s
TClingCallFunc::IFacePtr	1229.920s	0s	0s	3169.468s
TClass::GetCheckSum	973.722s	0s	0s	2351.981s
TClass::GetClass	420.100s	0s	0s	1285.435s
TClass::GetBaseClassOffset	405.951s	0s	0s	1131.577s
Selected 1 row(s):				10569.771s

Data Of Interest (CPU Metrics)

Viewing 0 of 0 selected stack(s)

No stack information

Wait Time

Primary cause of waiting is from ROOT locks

Event output time due to ROOT locks

Thread: TBB Worker ... (×7), cmsRunGlib...

Filter: 100.0% is shown | Process: [100.0%] cmsRunGlibC | Thread: Any Thread | Module: Any Module | Utilization: Any Utilization

Initial Findings

CMS

Waiting on TFormula

A new TFormula made each time a jet correction was called

Waiting on ROOT calls from cut parser

Calling functions via the ROOT interface used always took a lock

Waiting on output

1/2 time waiting to talk to a particular instance of output module

1/2 minutes waiting on ROOT lock during TTree::Fill

Vast majority from TClass::GetCheckSum

Philippe Canal made changes to ROOT to reduce lock use

TFormula copying should avoid taking locks

Calling TTree::Fill will avoid most locks

Decreasing the cost of using ROOT to find base class offsets

Post ROOT Changes

<no current project> - Intel VTune Amplifier (on cmsdev02.cern.ch)

Welcome | r000cc

Concurrency Hotspots by CPU Usage viewpoint (change) — Intel VTune Amplifier XE 2016

Collection Log | Analysis Target | Analysis Type | Summary | Bottom-up | Caller/Callee | Top-down Tree | Platform

Grouping: Function / Call Stack

Function / Call Stack	CPU Time: Effective Time by Utilization (Idle, Poor, Ok, Ideal, Over)	Spin Time	Ove.. Time	Wait Time by Utilization (Idle, Poor, Ok, Ideal, Over)
TObjArray::Delete	1887.904s	0s	0s	8335.941s
TFormula::ClearFormula	1887.704s	0s	0s	8335.456s
TObjArray::~TObjArray	0.200s	0s	0s	0.485s
TClingCallFunc::IFacePtr	2133.122s	0s	0s	7844.491s
full_read.constprop.172				4294.434s
XrdSysTimer::Wait	0.100s	0s	0s	4293.703s
XrdSys::IOEvents::PollE::Begin	0.130s	0s	0s	4291.905s
TClass::GetClass	768.500s	0s	0s	3238.035s
TFormula::~TFormula	329.205s	0s	0s	1493.230s
Selected 1 row(s):	1887.904s	0s	0s	8335.941s

Data Of Interest (CPU Metrics)

Viewing 1 of 347 selected stack(s)

8.1% (167.105s of 2063.404s)

- libCore.so!TObjArr...elete - [Unknown]
- libHist.so!TFormu...known]:[Unknown]
- libHist.so!TFormu...known]:[Unknown]
- libHist.so!TFormu...known]:[Unknown]
- libCondFormatsJ...nown]:[Unknown]
- libCondFormatsJ...nown]:[Unknown]
- libCondFormatsJe...nown]:[Unknown]
- libJetMETCorrect...known]:[Unknown]
- libJetMETCorrect...known]:[Unknown]
- pluginJetMETCorr...nown]:[Unknown]
- pluginDQMOffline...nown]:[Unknown]

Timeline: 500s 1000s 1500s 2000s 2500s 3000s 3500s 4000s 4500s

Thread: TBB Worker ... (×7), cmsRunGlib... — paused

Thread legend: Running, Waits, CPU Time, Spin and ..., CPU Sample, Transitions, CPU Usage, Thread Concurre...

No filters are applied. Process: Any Process | Thread: Any Thread | Module: Any Module | Utilization: Any Utilization

Post ROOT Changes

<no current project> - Intel VTune Amplifier (on cmsdev02.cern.ch)

Concurrency Hotspots by CPU Usage viewpoint (change)   Intel VTune Amplifier XE 2016

Collection Log | Analysis Target | Analysis Type | Summary | Bottom-up | Caller/Callee | Top-down Tree | Platform

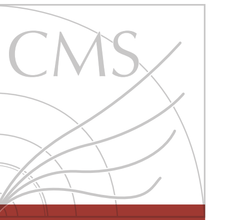
Grouping: Function / Call Stack

Function / Call Stack	CPU Time: Effective Time by Utilization	Spin Time	Overhead Time	Wait Time by Utilization
TObjArray::Delete	1887.904s	0s	0s	8335.941s
TFormula::ClearFormula	1887.704s	0s	0s	8335.456s
TObjArray::~TObjArray	0.200s	0s	0s	0.485s
TClingCallFunc::IFacePtr	2133.122s	0s	0s	7844.491s
full_read.constprop.172				4294.434s
XrdSysTimer::Wait	0.100s	0s	0s	4293.703s
XrdSys::IOEvents::PollE::Begin	0.130s	0s	0s	4291.905s
TClass::GetClass	768.500s	0s	0s	3238.035s
TFormula::~TFormula	329.205s	0s	0s	1493.230s
Selected 1 row(s):	1887.904s	0s	0s	8335.941s

Wait Time

TFormula and cling still a problem

Post ROOT Findings



Cut parser changes needed

- Philippe Canal added API to TMethod to allow caching of function pointer from cling

TFormula changes were insufficient

- Problem mitigated in ROOT 6.04 with new TFormula implementation
- Replaced with a hand made parser and executor
  - handles a subset of TFormula expressions
  - fully stateless parser and executor so extremely thread efficient
  - small memory footprint

First CMSSW Changes

Better Thread Utilization

Longer periods of CPU time

Function / Call Stack	CPU Time: Effective Time by Utilization	Spin Time	Overhead Time	Wait Time by Utilization
XrdCl::JobManager::RunJobs				
full_read.constprop.172				
XrdSysTimer::Wait				
XrdSys::IOEvents::PollE::Begin				
edm::one::OutputModuleBase::doEvent	0.200s	0s	0s	1056.987s
TClass::GetClass	86.359s	0s	0s	151.194s
[TBB worker]				96.649s
edm::eventsetup::DataProxy::get	2.300s	0s	0s	84.834s
edm::SharedResourcesAcquirer::lock	0.100s	0s	0s	52.186s
Selected 1 row(s):	0.200s	0s	0s	1056.987s

First CMSSW Changes

<no current project> - Intel VTune Amplifier (on cmsdev02.cern.ch)

Concurrency Hotspots by CPU Usage viewpoint (change)   Intel VTune Amplifier XE 2016

Collection Log | Analysis Target | Analysis Type | Summary | Bottom-up | Caller/Callee | Top-down Tree | Platform

Grouping: Function / Call Stack

Function / Call Stack	CPU Time: Effective Time by Utilization	Spin Time	Overhead Time	Wait Time by Utilization
XrdCl::JobManager::RunJobs				8203.580s
full_read.constprop.172				2734.600s
XrdSysTimer::Wait	0.200s	0s	0s	2734.473s
XrdSys::IOEvents::PollE::Begin	0.700s	0s	0s	2730.719s
edm::one::OutputModuleBase::doEvent	0.200s	0s	0s	1056.987s
TClass::GetClass	86.359s	0s	0s	151.194s
[TBB worker]				96.649s
edm::eventsetup::DataProxy::get	2.300s	0s	0s	84.834s
edm::SharedResourcesAcquirer::lock	0.100s	0s	0s	52.186s
Selected 1 row(s):	0.200s	0s	0s	1056.987s

Wait Time

Primarily waiting to run an instance of output module

First CMSSW Changes

<no current project> - Intel VTune Amplifier (on cmsdev02.cern.ch)

Welcome | r002cc

Concurrency Hotspots by CPU Usage viewpoint (change) — Intel VTune Amplifier XE 2016

Collection Log | Analysis Target | Analysis Type | Summary | Bottom-up | Caller/Callee | Top-down Tree | Platform

Grouping: Function / Call Stack

Function / Call Stack	CPU Time: Effective Time by Utilization (Idle, Poor, Ok, Ideal, Over)	CPU Time: Spin Time	CPU Time: Ove.. Time	Wait Time by Utilization (Idle, Poor, Ok, Ideal, Ove)
tbb::internal::allocate_root_proxy::allocate	3337.012s	0s	0s	
_ZN3edm15SerialTaskQueue11pushAndWaitIZNK12_GL	3337.012s	0s	0s	
operator new	472.198s	0s	0s	
operator delete	295.489s	0s	0s	
func@0x2e40	291.564s	0s	0s	
func@0x11c20	204.431s	0s	0s	
atan2f	135.800s	0s	0s	
std::__ostream_insert<char, std::char_traits<char>>	124.290s	0s	0s	17.718s
log	123.085s	0s	0s	
Selected 1 row(s):	3337.012s	0s	0s	

Data Of Interest (CPU Metrics)

Viewing 1 of 10 selected stack(s)

26.7% (891.418s of 3337.012s)

- libtbb.so.2!tbb::int...allocate - task.cpp
- pluginRecoPixelV...nown]:[Unknown]
- pluginRecoPixelV...nown]:[Unknown]
- libFWCoreFramew...nown]:[Unknown]
- libFWCoreFramew...nown]:[Unknown]
- libFWCoreFramew...nown]:[Unknown]
- libFWCoreFramew...nown]:[Unknown]
- libFWCoreFramew...nown]:[Unknown]
- libFWCoreFramew...nown]:[Unknown]
- libFWCoreFramew...nown]:[Unknown]
- libFWCoreFramew...nown]:[Unknown]

Timeline: 600s 800s 1000s 1200s 1400s 1600s 1800s 2000s 2200s 2400s 2600s 2800s 3000s 3200s

Thread: cmsRunGlib... | TBB Worker ... | TBB Worker ... | TBB Worker ... | TBB Worker ... | TBB Worker ... | TBB Worker ... | TBB Worker ...

Thread: Running, Waits, CPU Time, Spin and ..., CPU Sample, Transitions, CPU Usage, Thread Concurre...

Filter: 99.3% is shown | Process: Any Process | Thread: Any Thread | Module: Any Module | Utilization: Any Utilization

CPU Time

Lots of time in TBB internal

Findings

TBB time from calls to SerialTaskQueue

- CMS class meant to protect a resource from simultaneous access

SerialTaskQueue inappropriately used in an 'event' data class

- Class was doing a lazy evaluation of elements in its container
- First request for each item put task into the SerialTaskQueue
- Code wasn't actually thread safe anyway
  - other pieces of code were accessing the container outside of the SerialTaskQueue

Change: remove the use of SerialTaskQueue

- Removing the lazy evaluation had no impact on performance
  - the code had been prematurely optimized

Final Results

<no current project> - Intel VTune Amplifier (on cmsdev02.cern.ch)

Welcome | r003cc

Concurrency Hotspots by CPU Usage viewpoint (change) — Intel VTune Amplifier XE 2016

Collection Log | Analysis Target | Analysis Type | Summary | Bottom-up | Caller/Callee | Top-down Tree | Platform

Grouping: Function / Call Stack

Function / Call Stack	CPU Time: Effective Time by Utilization (Idle, Poor, Ok, Ideal, Over)	Spin Time	Ove.. Time	Wait Time by Utilization (Idle, Poor, Ok, Ideal, Over)
edm::one::OutputModuleBase::doEvent	0.100s	0s	0s	2556.050s
TClass::GetClass	154.798s	0s	0s	195.946s
[TBB worker]				77.842s
IOChannel::write	1.991s	0s	0s	52.544s
edm::SharedResourcesAcquirer::lock	0.213s	0s	0s	36.348s
std::__ostream_insert<char, std::char_traits<char>>	101.317s	0s	0s	22.423s
TClass::GetStreamerInfo	15.782s	0s	0s	20.349s
std::condition_variable::wait				19.880s
Selected 1 row(s):	0.100s	0s	0s	2556.050s

Data Of Interest (CPU Metrics)
Viewing 1 of 2 selected stack(s)
50.0% (0.100s of 0.200s)
libFWCoreFramewo...ent - [Unknown]
libFWCoreFramew...nown]:[Unknown] (×9)

Timeline: 600s 700s 800s 900s 1000s 1100s 1200s 1300s 1400s 1500s 1600s 1700s 1800s 1900s 2000s 2100s 2200s
Thread: TBB Worker ... (×7), _start (TID: 8...
Legend: Thread — Running, Waits, CPU Time, Spin and ..., CPU Sample, Transitions; CPU Usage; Thread Concurre...

Filter: 99.4% is shown | Process: [100.0%] cmsRunGlibC | Thread: Any Thread | Module: Any Module | Utilization: Any Utilization

Final Results

Good Thread Utilization

High continuous thread utilization

Overall job time is much shorter

Final Results

<no current project> - Intel VTune Amplifier (on cmsdev02.cern.ch)

Concurrency Hotspots by CPU Usage viewpoint (change) — Intel VTune Amplifier XE 2016

Collection Log | Analysis Target | Analysis Type | Summary | Bottom-up | Caller/Callee | Top-down Tree | Platform

Grouping: Function / Call Stack

Function / Call Stack	CPU Time: Effective Time by Utilization	Spin Time	Ove.. Time	Wait Time by Utilization
edm::one::OutputModuleBase::doEvent	0.100s	0s	0s	2556.050s
TClass::GetClass	154.798s	0s	0s	195.946s
[TBB worker]				77.842s
IOChannel::write	1.991s	0s	0s	52.544s
edm::SharedResourcesAcquirer::lock	0.213s	0s	0s	36.348s
std::__ostream_insert<char, std::char_traits<char>>	101.317s	0s	0s	22.423s
TClass::GetStreamerInfo	15.782s	0s	0s	20.349s
std::condition_variable::wait				19.880s
Selected 1 row(s):	0.100s	0s	0s	2556.050s

Wait Time

Primarily waiting to run an instance of output module

Filter: 99.4% is shown | Process: [100.0%] cmsRunGlibC | Thread: Any Thread | Module: Any Module | Utilization: Any Utilization

Final Results

<no current project> - Intel VTune Amplifier (on cmsdev02.cern.ch)

Click to view your appointments and tasks

Welcome | r003cc

Concurrency Hotspots by CPU Usage viewpoint (change)

Intel VTune Amplifier XE 2016

Collection Log | Analysis Target | Analysis Type | Summary | Bottom-up | Caller/Callee | Top-down Tree | Platform

Grouping: Function / Call Stack

Function / Call Stack	CPU Time: Effective Time by Utilization (Idle, Poor, Ok, Ideal, Over)	Spin Time	Ove.. Time	Wait Time by Utilization (Idle, Poor, Ok, Ideal, Over)
func@0x11c20	451.567s	0s	0s	
operator new	399.706s	0s	0s	
func@0x2e40	263.205s	0s	0s	
operator delete	236.299s	0s	0s	
TClass::GetClass	154.798s	0s	0s	195.946s
func@0x11fb0	119.990s	0s	0s	
atan2f	119.099s	0s	0s	
std::__ostream_insert<char, std::char_traits<char>>	101.317s	0s	0s	22.423s
log	93.845s	0s	0s	
Selected 1 row(s):	83.401s	0s	0s	

Data Of Interest (CPU Metrics)

Viewing 1 of 603 selected stack(s)

1.3% (1.100s of 83.401s)

libMagneticFieldP...tBxyz - [Unknown]
libMagneticField...known]:[Unknown]
libMagneticField...known]:[Unknown]
libTrackingToolsT...known]:[Unknown]
pluginRecoTracke...nown]:[Unknown]
pluginRecoTracke...known]:[Unknown]
libTrackingTools...known]:[Unknown]
pluginRecoTracke...nown]:[Unknown]
pluginRecoTracke...nown]:[Unknown]
pluginRecoTracke...nown]:[Unknown]
pluginRecoTracke...nown]:[Unknown]

600s 700s 800s 900s 1000s 1100s 1200s 1300s 1400s 1500s 1600s 1700s 1800s 1900s 2000s 2100s 2200s

Thread: TBB Worker ... (×7), _start (TID: 8...

Thread: Running, Waits, CPU Time, Spin and ..., CPU Sample, Transitions; CPU Usage; Thread Concurre...

CPU Time

func@0X* are all compression calls

Filter: 99.4% is shown | Process: [100.0%] cmsRunGlibC | Thread: Any Thread | Module: Any Module | Utilization: Any Utilization

Next Step

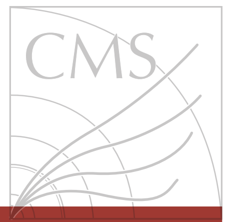
Full CMSSW threading framework implementation

- Switching to TBB task per module will work around output module
  - If event 1 is running the output module event 2 could run a different module

Work on output module to make more thread efficient

- Decrease time in a single thread
- Look into utilizing multiple threads when compressing buffers

VTune Calibration



The value VTune gives for 'concurrency' does not correspond to timing measurements of program

VTune much more pessimistic

Increased VTune concurrency does correspond to real world speedups

Program Changes	Speedup	Regular Job Time (s)	VTune Concurrency	VTune Job Time (s)
Initial	5.5	1826	2.1	16440
ROOT	5.7	1748	2.6	9884
CMSSW 1	6.1	1641	7.2	3363
Final	6.2	1605	5.0*	2429

* fraction of the job was in the end of job single threaded part. GUI histogram of thread utilizations showed most time spent at 7 cores active with second most at 1.

Conclusion

CMS

VTune is a useful tool

- It is capable of running on HEP frameworks
- It does identify code that hampers threading performance

Not without its problems

- Sometimes fails to report work done on some threads
- Data gathering appears to strongly affect job running time and distort findings

The problems are not sufficient to stop me from using VTune