ITDevCon
European Delphi Conference

6, 7 october 2016 - ROMA

How I Learned to Love FastMM Internals

Primož Gabrijelčič

http://primoz.gabrijelcic.org

bit Time professionals
consulenza | formazione | sviluppo

ITDevCon
European Delphi Conference

History

History

- Developed by Pierre LeRiche for the FastCode project
  - https://en.wikipedia.org/wiki/FastCode
  - Version 4, hence FastMM4
- Included in RAD Studio since version 2006
  - http://www.tindex.net/Language/FastMMmemorymanager.html
- Much improved since
  - Don’t use default FastMM, download the fresh one
  - https://github.com/pleriche/FastMM4

Features

- Fast
- Fragmentation resistant
- Access to > 2GB
- Simple memory sharing
- **Memory leak reporting**
- **Catches some memory-related bugs**

Problems

- Can get slow in multithreaded environment
- Can get VERY slow in multithreaded environment

ITDevCon
European Delphi Conference

FastMM4 Internals

Top View

- Three memory managers in one

- **Small blocks** (< 2,5 KB)
  - Most frequently used (99%)
  - Medium blocks, subdivided into small blocks
- **Medium blocks** (2,5 – 260 KB)
  - Allocated in chunks (1,25 MB) and subdivided into lists
- **Large blocks** (> 260 KB)
  - Allocated directly by the OS

Details

- One large block allocator
- One medium block allocator
- Multiple (54+2) small block allocators
  - `SmallBlockTypes`
  - Custom, optimized Move routines (FastCode)

- Each allocator has its own lock
  - If SmallAllocator is locked, SmallAllocator+1 or SmallAllocator+2 is used

Problem

- Multithreaded programs are slow?
- Threads are fighting for allocators.
- Easy to change the program to bypass the problem.
  - Well, sometimes.
- Hard to find out the responsible code.

Demo

- Steve Maughan: http://www.stevemaughan.com/delphi/delphi-parallel-programming-library-memory-managers/

- http://www.thedelphigeek.com/2016/02/finding-memory-allocation-bottlenecks.html

ITDevCon
European Delphi Conference

Diagnosing FastMM4 Bottlenecks

$DEFINE LogLockContention

FastMM4 Locking

```
if IsMultiThread then begin
  while LockCmpxchg(0, 1, @MediumBlocksLocked) <> 0 do begin
{$ifdef NeverSleepOnThreadContention}
{$ifdef UseSwitchToThread}
    SwitchToThread; //any thread on the same processor
{$endif}
{$else}
    Sleep(InitialSleepTime); // 0; any thread that is ready to run
    if LockCmpxchg(0, 1, @MediumBlocksLocked) = 0 then
      Break;
    Sleep(AdditionalSleepTime); // 1; wait
{$endif}
  end;
end;
```

Lock Contention Logging

```
LockMediumBlocks({$ifdef LogLockContention}LDidSleep{$endif});

{$ifdef LogLockContention}
if LDidSleep then
  ACollector := @MediumBlockCollector;
{$endif}

if Assigned(ACollector) then begin
  GetStackTrace(@LStackTrace, StackTraceDepth, 1);
  MediumBlockCollector.Add(@LStackTrace[0], StackTraceDepth);
end;
```

FastMM4DataCollector

- Opaque data
- Completely static
  - Can’t use MM inside MM
  - Agreed max data size
- Most Frequently Used
- Generational
  - Reduce the problem of local maxima
  - Two generations, sorted
    - 1024 slots in Gen1
    - 256 slots in Gen2
  - Easy to expand to more generations

Output

- Results for all allocators are merged

```
LargeBlockCollector.GetData(mergedData, mergedCount);
MediumBlockCollector.GetData(data, count);
LargeBlockCollector.Merge(mergedData, mergedCount, data, count);
for i := 0 to High(SmallBlockTypes) do begin
  SmallBlockTypes[i].BlockCollector.GetData(data, count);
  LargeBlockCollector.Merge(mergedData, mergedCount, data, count);
end;
```

- Top 10 “call sites” are written to `<programname>_MemoryManager_EventLog.txt`

ITDevCon
European Delphi Conference

Findings

It is hard to release memory

- Time is mostly wasted in FreeMem
- GetMem (with small blocks) can “upgrade” to unused allocator
  - One thread doesn’t block another
- FreeMem must work with the allocator that “produced” the memory
  - One thread blocks another

ITDevCon
European Delphi Conference

Solution

Solution

- If allocator is locked, delay the FreeMem
- Memory block is pushed on a ‘to be released’ list
- Each allocator gets its own “release stack”

```
while LockCmpxchg(0, 1, @LPSmallBlockType.BlockTypeLocked) <> 0 do begin
{$ifdef UseReleaseStack}
  LPReleaseStack := @LPSmallBlockType.ReleaseStack;
  if (not LPReleaseStack^.IsFull) and LPReleaseStack^.Push(APointer) then begin
    Result := 0;
    Exit;
  end;
{$endif}
```

- When allocator is successfully locked, all memory from its release stack is released.

FastMM4LockFreeStack

- Very fast lock-free stack implementation
  - Taken from OmniThreadLibrary
- Windows only
- Dynamic memory
  - Uses HeapAlloc for memory allocation

Problems

- Release stacks work, but not perfectly

1. FreeMem can still block if multiple threads are releasing similarly sized memory blocks.
   - Solution: Hash all threads into a pool of release stacks.

2. Somebody has to clean after terminated threads.
   - Solution: Low-priority memory release thread.
   - Currently only for medium/large blocks.
   - CreateCleanupThread/DestroyCleanupThread

Bunch of release stacks

```
while LockCmpxchg(0, 1, @LPSmallBlockType.BlockTypeLocked) <> 0 do begin
{$ifdef UseReleaseStack}
  LPReleaseStack := @LPSmallBlockType.ReleaseStack[GetStackSlot];
  if (not LPReleaseStack^.IsFull) and LPReleaseStack^.Push(APointer) then
  begin
    Result := 0;
    Exit;
  end;
{$endif}
```

- GetStackSlot hashes thread ID into [0..NumStacksPerBlock-1] range

Danger, Will Robinson!

- Used in production
  - Still, use with care
- Incompatible with FullDebugMode
- $DEFINE UseReleaseStack

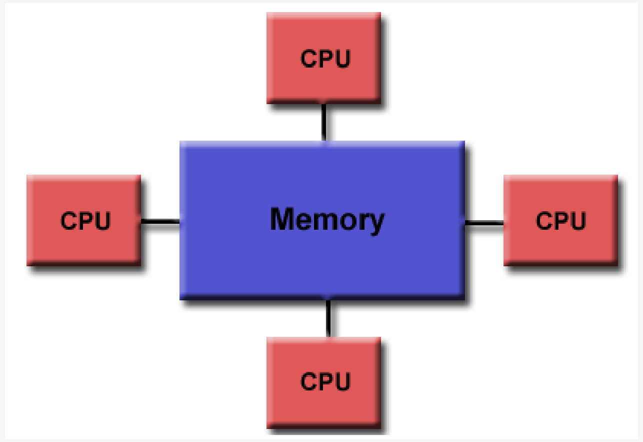
ITDevCon
European Delphi Conference

NUMA

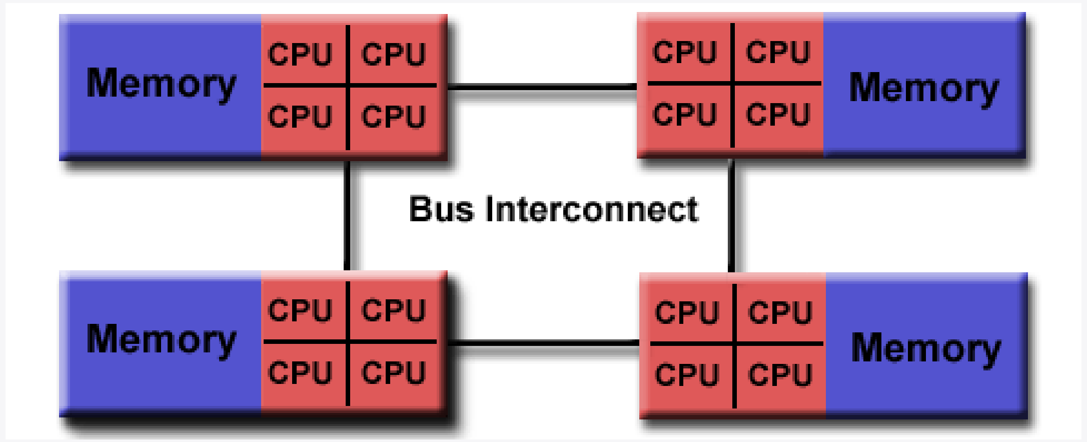
Non-Uniform Memory Access

Non-Uniformed Memory Access

SMP



NUMA



Source: *Introduction to Parallel Computing*, https://computing.llnl.gov/tutorials/parallel_comp/

NUMA brings problems

- Different “cost” for memory access

	00	01	02	03
00	1.0	1.6	1.9	3.4
01	1.8	1.9	2.2	3.5
02	2.1	2.2	1.8	2.6
03	2.2	3.1	2.8	2.1

- Measurement from a real system
  - 80 cores, 20 in each NUMA node
  - Coreinfo, Mark Russinovich
    - Not very accurate measurement

Solution

- Node-local memory allocation

- FastMM implementation: per-node allocators
- https://github.com/gabr42/FastMM4-MP/tree/numa

- **VERY** experimental!

Solution, part 2

- How to use more than 64 cores in your program?

- OmniThreadLibrary with NUMA extensions
  - https://github.com/gabr42/OmniThreadLibrary/tree/numa
  - Environment.ProcessorGroups, Environment.NUMANodes
  - IOmniTaskControl.ProcessorGroup, IOmniTaskControl.NUMANode
  - IOmniThreadPool.ProcessorGroups, IOmniThreadPool.NUMANodes

“NUMA” for Developers

- `bcdedit /set groupsize 2`
  - https://msdn.microsoft.com/en-us/library/windows/hardware/ff542298(v=vs.85).aspx

ITDevCon
European Delphi Conference

Questions?