NFD - Task #3610

FIB+PIT profiling

04/28/2016 12:48 PM - Junxiao Shi

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Chengyu Fan	**% Done:**	100%
Category:		**Estimated time:**	12.00 hours
Target version:	v0.5		

Description

Profile the performance of FIB and PIT, and understand where the bottlenecks are.

Related issues:

Blocked by NFD - Feature #3571: FIB+PIT benchmark	**Closed**	**03/28/2016**

History

#1 - 04/28/2016 12:49 PM - Junxiao Shi

- Blocked by Feature #3571: FIB+PIT benchmark added

#2 - 04/28/2016 11:44 PM - Zhuo Li

- Start date set to 04/29/2016

#3 - 07/07/2016 03:35 PM - Zhuo Li

- Status changed from New to In Progress

#4 - 07/22/2016 11:15 AM - Junxiao Shi

- Status changed from In Progress to New

- Assignee deleted (Zhuo Li)

- Start date deleted (04/29/2016)

Beichuan says Zhuo will not work on this recently.

#5 - 07/23/2016 03:02 AM - Junxiao Shi

- Assignee set to Chengyu Fan

Chengyu agrees to work on this issue but he cannot start until August.

#6 - 08/29/2016 02:52 PM - Chengyu Fan

- File callgrind.pit-fib-performance-profiling.out added

- % Done changed from 0 to 100

I have run the pit_fib performance test on an ONL Host8core machine. The Callgrind profiling output file is uploaded. Please use kcachegrind to open it for details.

The pit-fib-benchmark.cpp simulates FIB and PIT operations on one million Interest-Data exchanges. It generates packets and populates FIB before the actual tests. The tests include finding longest prefix match in FIB, finding matched Data in PIT and erasing PIT entries.

The results show that the FIB and PIT tests take 11374372 microseconds. The whole pit-fib-benchmark takes 25.655s. This means more than half of the time is used by the generatePacketsAndPopulateFib() function. FIB and PIT operations (exclude generatePacketsAndPopulateFib()) can achieve 87,950 Interest-Data exchanges per second.

More information is given below:

```
(1) generatePacketsAndPopulateFib() takes 49.15% of the running time.
     Its child, the ndn::Name::Name constructor uses around 26.06%. The other child nfd::fib::insert() takes 8
.74%

(2) Pit as a whole takes 44.46%
```

```
     nfd::Pit::findOrInsert() takes 19.92%
     nfd::Pit::findAllDataMatches() takes 16.37%
     nfd::Pit::erase() uses the other 8.32%

(3) the most time-consuming ndn function is ndn::Name::wireEncode() 21.51%.
     It is called by both nfd::name_tree::computeHash() 14.00% and nfd::name_tree::computeHashSet() 7.52%

(4) NameTree:
    a. the most time-consuming NameTree function is nfd::NameTree::lookup() 26.01%.
       The Fib::insert() contributes 8.74% (called by generatePacketsAndPopulateFib()). The rest is from Pit::
findOrInsert() 19.92%.

    b. another time-consuming function is nfd::NameTree::findAllMatches() 12.90%. They are from nfd::Pit::find
AllDataMatches()
```

Since the ndn::Name::wireEncode() alone uses 21.51% of time, I have tried to use TCMalloc library to optimize the memory management. The results confirmed my assumption. The FIB and PIT operations go down to 8499654 microseconds. This means FIB and PIT operations can achieve 117,647 Interest-Data exchanges per second.

#7 - 08/29/2016 04:02 PM - Junxiao Shi

- Status changed from New to Feedback

I should have said this earlier, but generatePacketsAndPopulateFib should not be part of the profiling: it's the initial setup.

Can you report only the portion between two time::steady_clock::now() calls?

https://github.com/named-data/NFD/blob/dbef6dc8e2e32160b9e826ca0f32b9a00bb759df/tests/other/pit-fib-benchmark.cpp#L118-L133

#8 - 08/29/2016 04:25 PM - Davide Pesavento

Please provide the exact versions of NFD and ndn-cxx (git commit SHA) used to perform the test in note-6.

#9 - 08/29/2016 04:41 PM - Chengyu Fan

The number I reported is the time between two time::steady_clock::now() calls.

Since I have trouble to install openssl library for ndn-cxx on ONL, I used the older version. The ndn-cxx is 0.4.1 (commit SHA = 2e52d7cab4e03631db7f6c89a2daf7de590dd29e). The NFD is 4f1afaca16ba02b40651f62e0f28e56d0b7cbf9f, which is the commit for face system benchmark, but I modified the pit-fib-benchmark.cpp code as https://gerrit.named-data.net/#/c/3021/.

Please let me know if I chose the wrong version.

Without TCMalloc, the time between two time::steady_clock::now() calls is 11374372 microseconds. With TCMalloc, the time is 8499654 microseconds.

#10 - 08/29/2016 05:05 PM - Junxiao Shi

It's necessary to rerun the profiling with latest version, because I have changed NameTree significantly.

The profiling should only report function calls between two nows; it should exclude anything under initial setup procedure.

#11 - 08/30/2016 02:20 PM - Chengyu Fan

- File callgrind.pit-fib-performance-profiling.out added

- File call-graph-for-pit-fib-operations.png added

The profiling results for the latest NFD (9685cc5682c9e1d642747649294f469e316733ab) is uploaded. The ndn-cxx version used for this test is 667370f102f7953968a6d0d60a782cf0ef33d9d5. The profiling is running for the portion between two time::steady_clock::now() calls. Please refer to the uploaded file call-graph-for-pit-fib-operations.png for the call graph.

```
(1) nfd::Pit::findOrInsert() takes 42.33% of the running time

(2) nfd::Pit::findAllDataMatches() takes 39.51%

(3) nfd::Pit::erase() uses the other 16.23%

(4) ndn::Name::wireEncode() uses 40.27%, and it is called by nfd::name_tree::computesHash()
```

To measure the time between two time::steady_clock::now() calls, I run 10 experiments for both without TCMalloc and with TCMalloc.
Without TCMalloc, the time is in range (7833889, 10918891) microseconds, and the median is 8606859.5 microseconds. With TCMalloc, the time is in range (6012537, 8827668) microseconds, and the median is 6935110.5 microseconds.

#12 - 08/30/2016 06:23 PM - Davide Pesavento

Chengyu Fan wrote:

> The profiling is running for the portion between two time::steady_clock::now() calls.

How did you do that? Did you put CALLGRIND_{START,STOP}_INSTRUMENTATION macros in the code?

#13 - 08/30/2016 07:29 PM - Chengyu Fan

Davide Pesavento wrote:

> Chengyu Fan wrote:
>
> > The profiling is running for the portion between two time::steady_clock::now() calls.
>
> How did you do that? Did you put CALLGRIND_{START,STOP}_INSTRUMENTATION macros in the code?

Yes, I added these macros in the pit-fib-benchmark.cpp code. Please let me know if there are any issues

#14 - 08/31/2016 12:49 AM - Davide Pesavento

That's good, it's what I had in mind. We should probably add those calls to the committed benchmark code for posterity's sake. I'll prepare a patch.

#15 - 08/31/2016 08:09 PM - Davide Pesavento

https://gerrit.named-data.net/3155

#16 - 09/01/2016 12:48 AM - Davide Pesavento

Junxiao asks on gerrit:

> Can you post on Redmine the command line to generate a callgrind report? I need to test this.

$ valgrind --tool=callgrind --instr-atstart=no ./build/whatever

#17 - 01/25/2018 01:11 PM - Davide Pesavento

- Category deleted (Integration Tests)

- Status changed from Feedback to Closed

Files

callgrind.pit-fib-performance-profiling.out	586 KB	08/29/2016	Chengyu Fan
callgrind.pit-fib-performance-profiling.out	66.1 KB	08/30/2016	Chengyu Fan
call-graph-for-pit-fib-operations.png	175 KB	08/30/2016	Chengyu Fan