Optimization Guide

IPFS Playbook on 3rd Generation Intel® Xeon® Scalable Processors & Intel® Optane™ Persistent Memory

Revise

V0.3	1. change DDR4 lookahead to 300 and keep bps as default 800	
V0.4	1. add three variables and change performance recommendation.	
V0.6	1. add recommendation of SWAP. 2. add throughput per day	
V0.7	1. simply DRAM and BPS optimization method. 2, add sha256 optimization	

1. Lotus-Bench 安装

安装 Lotus-Bench

https://docs.filecoin.io/get-started/lotus/installation/#software-dependencies

Build and install Lotus:

```
export RUSTFLAGS="-C target-cpu=native -g"
export FFI_BUILD_FROM_SOURCE=1
```

*Make clean all
*Make install

https://docs.filecoin.io/mine/lotus/benchmarks/

*Make lotus-bench

2. 性能优化

Lotus-Bench安装完成后，需要采用英特尔优化后的代码，以及相应的BIOS和系统配置，才能获得性能提升。对于不同的内存（DRAM和英特尔傲腾持久内存）选择会有各自的优化配置方法，详见2.2和2.3.

2.1. 更新优化的代码

下载更新代码包lotus171-optimization-code.zip， 链接: https://pan.baidu.com/s/1d-b0bfoRUGOF4jGjOnEpDg 提取码: tvus

1. 更新代码包中的3个文件, 可以通过find 命令找到文件的位置， 例如 find / -name cores.rs * 确定storage_proof 版本 7.0.0?

2, 更新SHA256 library

a. Copy attached **libsimdsha2block.a** to **/usr/local/lib64/** Folder(if not existing, please create it);
b. Go to $HOME/.cargo/registry/src/github.com-1ecc6299db9ec823/**sha2-0.9.3**/ folder and
   a. apply the patch:
   git apply 0001-Integrate-optimized-SHA256-with-SIMD-and-expose-msg_.patch

```
root@filecoin-WilsonCity:/home/filecoin/.cargo/registry/src/github.com-1ecc6299db9ec823/sha2-0.9.3# cp
ilecoin/0001-Integrate-optimized-SHA256-with-SIMD-and-expose-msg_.patch ./
root@filecoin-WilsonCity:/home/filecoin/.cargo/registry/src/github.com-1ecc6299db9ec823/sha2-0.9.3# git
0001-Integrate-optimized-SHA256-with-SIMD-and-expose-msg_.patch
0001-Integrate-optimized-SHA256-with-SIMD-and-expose-msg_.patch:100: trailing whitespace.
    pub fn sha256_msg_schedule_x8(in1: *const u8, out1: *mut u8,
0001-Integrate-optimized-SHA256-with-SIMD-and-expose-msg_.patch:102: trailing whitespace.
                                  in3: *const u8, out3: *mut u8,
0001-Integrate-optimized-SHA256-with-SIMD-and-expose-msg_.patch:104: trailing whitespace.
                                  in5: *const u8, out5: *mut u8,
0001-Integrate-optimized-SHA256-with-SIMD-and-expose-msg_.patch:106: trailing whitespace.
                                  in7: *const u8, out7: *mut u8,
0001-Integrate-optimized-SHA256-with-SIMD-and-expose-msg_.patch:113: trailing whitespace.
    pub fn sha256_msg_schedule_x4(in1: *const u8, out1: *mut u8,
warning: squelched 4 whitespace errors
warning: 9 lines add whitespace errors.
```

b. Modify the **Cargo.toml** and add below line in the **"[package]"** section:

Add : build = "build.rs"

```
keywords = ["crypto", "sha2", "hash", "digest"]
categories = ["cryptography", "no-std"]
license = "MIT OR Apache-2.0"
repository = "https://github.com/RustCrypto/hashes"
build = "build.rs"
[dependencies.block-buffer]
version = "0.9"

[dependencies.cfg-if]
version = "1.0"
```

更新完成以后需要重新build

```
export RUSTFLAGS="-C target-cpu=native -g"
export FFI_BUILD_FROM_SOURCE=1
```

make clean all
make lotus-bench

更新成功以后，在lotus-bench的log文件中可以看到如下的信息

```
--This is Intel CPU , packagecore_count is :32, group_size is :3, group_count is :16 numa is:0

--This is Intel CPU , bps optimization is:1

 -- Intel CPU Core list : [[CoreIndex(0), CoreIndex(1), CoreIndex(1)], [CoreIndex(2), CoreIndex(3), CoreIndex(
3)], [CoreIndex(4), CoreIndex(5), CoreIndex(5)], [CoreIndex(6), CoreIndex(7), CoreIndex(7)], [CoreIndex(8), Co
reIndex(9), CoreIndex(9)], [CoreIndex(10), CoreIndex(11), CoreIndex(11)], [CoreIndex(12), CoreIndex(13), CoreI
ndex(13)], [CoreIndex(14), CoreIndex(15), CoreIndex(15)], [CoreIndex(16), CoreIndex(17), CoreIndex(17)], [Core
Index(18), CoreIndex(19), CoreIndex(19)], [CoreIndex(20), CoreIndex(21), CoreIndex(21)], [CoreIndex(22), CoreI
ndex(23), CoreIndex(23)], [CoreIndex(24), CoreIndex(25), CoreIndex(25)], [CoreIndex(26), CoreIndex(27), CoreIn
dex(27)], [CoreIndex(28), CoreIndex(29), CoreIndex(29)], [CoreIndex(30), CoreIndex(31), CoreIndex(31)]] , Let'
s Speed Up !!!
```

```
--This is Intel CPU , switch to use SIMD optimized SHA256

--This is Intel CPU , switch to use SIMD optimized SHA256
```

在代码中，增加了2个变量来控制相应的性能优化，请参考下面的解释。

export FIL_PROOFS_INTEL_CPU_SUPPORT=1	如果测试的是intel 平台，请打开下面的变量，开启专门针对intel 平台的优化功能
export FIL_PROOFS_INTEL_NUMA_NODE=1	如果想控制任务跑在特定的CPU上， 可以通过这个变量 0: balance tasks to all NUMA node. 1: bind tasks to NUMA1 2: bind tasks to NUMA2

建议在生产环境运行时，确保所有任务的内存占用不超过物理内存容量，如果实际的内存占用超过物理内存，确保将SWAP区放置在特别快速的NVMe SSD如Intel Optane SSD P5800X上，并且同数据盘分开放置。

2.2. *[Intel Icelake CPU] 性能优化建议 - Both DDR4 AND BPS Configuration

2.2.1. BIOS Knob :

Socket Configuration-> Advanced Power Management Configuration ->CPU C State Control ->Enable Monitor MWAIT **:<Disable>**

Socket Configuration-> Advanced Power Management Configuration ->CPU C State Control ->CPU C6 Report :**<Disable>**

Socket Configuration-> Advanced Power Management Configuration ->Package C State Control ->Package C State : **<C0/C1 state>**

Socket Configuration-> IIO Configuration->PCI-E ASPM Support :**<No>**

Socket Configuration-> Common RefCode Configuration->NUMA:**<Enable>**

Socket Configuration-> Processor Configuration-> Hyper-Threading [ALL]:**<Enable>**

Socket Configuration-> Memory Configuration->Pmem Configuration->PMem Performance Setting**:<Balanced Profile>**

2.2.2. Environment list

```
export FIL_PROOFS_MAXIMIZE_CACHING=1
export FIL_PROOFS_USE_MULTICORE_SDR=1
export FIL_PROOFS_MULTICORE_SDR_PRODUCERS=2
export FIL_PROOFS_MULTICORE_SDR_LOOKAHEAD=300
export FIL_PROOFS_INTEL_CPU_SUPPORT=1
```

以下变量取决于你是否用GPU 跑P2.

```
export FIL_PROOFS_USE_GPU_COLUMN_BUILDER=1
export FIL_PROOFS_USE_GPU_TREE_BUILDER=1
```

2.2.3. 运行测试

完成配置优化和代码优化后，可以通过以下命令执行测试

数据盘跨CPU 对性能有很大的影响， 例如拿INTEL Icelake 服务器举例， 每个服务器平台有2个CPU, 最好的配置是CPU1 上跑的lotus 任务访问CPU1 下挂的硬盘， CPU2上跑的lotus 任务访问CPU2 下挂的硬盘， 具体的操作方法如下：

以 2个Icelake CPU , 每个CPU 有1T 的内存， 所以每个CPU 跑7个lotus 任务为例。

```
export FIL_PROOFS_INTEL_NUMA_NODE=1   // 控制下面7个任务跑到CPU1 上
nohup ./lotus-bench sealing --sector-size=64GiB --storage-dir=/mnt/CPU1-SDD/  --num-sectors=7 --parallel=7 --skip-commit2 --skip-unseal  >> /mnt/CPU1-SDD/log/bench64G-7.log 2>&1 &
```

```
export FIL_PROOFS_INTEL_NUMA_NODE=2   // 控制下面7个任务跑到CPU2 上
nohup ./lotus-bench sealing --sector-size=64GiB --storage-dir=/mnt/CPU2-SDD/  --num-sectors=7 --parallel=7 --skip-commit2 --skip-unseal  >> /mnt/CPU2-SDD/log/bench64G-7.log 2>&1 &
```

2.2.4. Hardward Configuration :

CPU	6346 x 2	8358 x 2
Memory /BPS	32G -3200 X16 128G BPS x16	64G -3200 X16 256G BPS x16
HDD	>12T NVME SSD (RAID0) x2 (每个CPU 1个RAID) + 1 NVME for OS	>20T NVME SSD (RAID0) X2 (每个CPU 1个) + 1 NVME for OS
OS	Ubuntu 20.04 or 20.10	Lotus-bench 1.7.1
GPU	3090 X 1	3090 X 1

2.3. 性能参考数据

通过优化后，在英特尔实验室获得了以下性能数据。使用者请重点关注优化方法和代码，此数据仅作参考，具体测试中，配置不同，测试结果会有差异。

全部的性能数据 持续补充中.....

配置	任务数	时间
6346CPU+2T DDR4	1 32G 扇区	176分钟
6346CPU+2T DDR4	14 64G 扇区	380分钟
6346CPU+2T BPS (512G DDR4)		
8358CPU + 2T DDR4	1 32G 扇区	190分钟
8358CPU + 2T DDR4	30 32G 扇区	220分钟

8358CPU + 2T BPS(512G DDR4)		
8358CPU + 4T BPS(1T DDR4)	30 64G 扇区	446分钟（2个优化）

2.4. 算力参考数据

根据2.4章节的性能参考数据，单台服务器的算力估值如下图所示：

3. *推荐测试方法

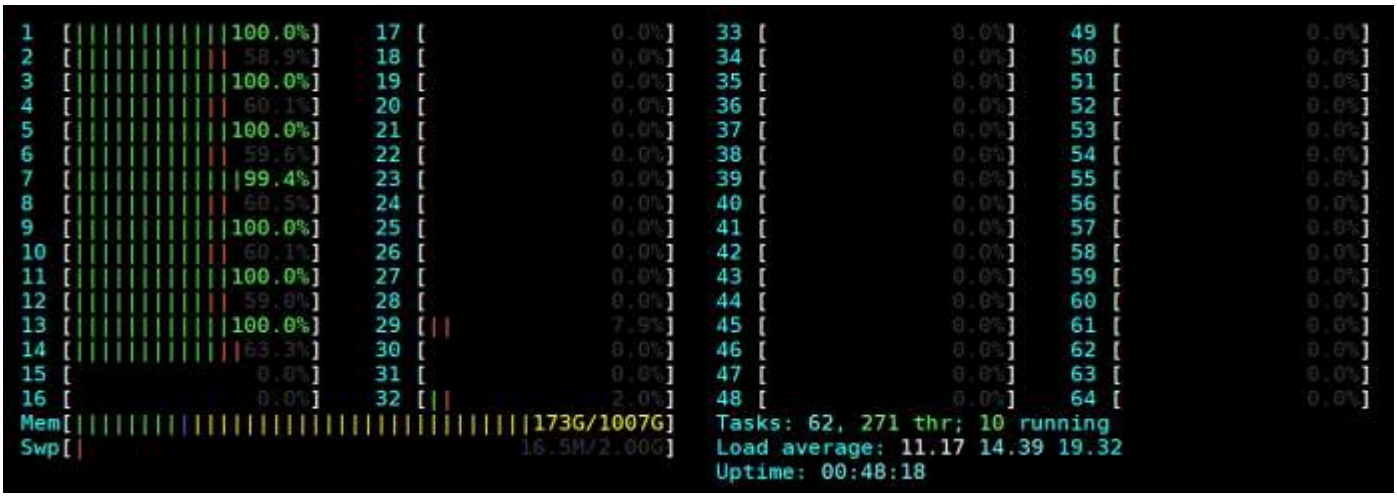
数据盘跨CPU 对性能有很大的影响， 例如拿INTEL Icelake 服务器举例， 每个服务器平台有2个CPU, 最好的配置是CPU1 上跑的lotus 任务访问CPU1 下挂的硬盘， CPU2上跑的lotus 任务访问CPU2 下挂的硬盘， 具体的操作方法如下：

以 2个Icelake CPU , 每个CPU 有1T 的内存， 所以每个CPU 跑7个lotus 任务为例。

```
export FIL_PROOFS_INTEL_NUMA_NODE=1   // 控制下面的任务跑到CPU1 上
nohup ./lotus-bench sealing --sector-size=64GiB --storage-dir=/mnt/CPU1-SDD/  --num-sectors=7 --parallel=7
--skip-commit2 --skip-unseal  >> /mnt/CPU1-SDD/log/bench64G-7.log 2>&1 &
```

```
export FIL_PROOFS_INTEL_NUMA_NODE=2   // 控制下面的任务跑到CPU2 上
nohup ./lotus-bench sealing --sector-size=64GiB --storage-dir=/mnt/CPU2-SDD/  --num-sectors=7 --parallel=7
--skip-commit2 --skip-unseal  >> /mnt/CPU2-SDD/log/bench64G-7.log 2>&1 &
```

4. DEBUG 方法 -TBD



Notices & Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Your costs and results may vary.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not "commercial" names and not intended to function as trademarks

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.