# SFG

* 1 [Introduction](#SFG-Introduction)
* 2 [Goal](#SFG-Goal)
  + 2.1 [Llama 70B Prompt Stage SFG with torch.compile](#SFG-Llama70BPromptStageSFGwithtorch.com)
    - 2.1.1 [Status: SFG with torch.compile](#SFG-Status:SFGwithtorch.compile)
    - 2.1.2 [Llama config for SFG:](#SFG-LlamaconfigforSFG:)
    - 2.1.3 [Prompt stage tensor shapes:](#SFG-Promptstagetensorshapes:)
    - 2.1.4 [Projection:](#SFG-Projection:)
    - 2.1.5 [Design considerations from Deepak:](#SFG-DesignconsiderationsfromDeepak:)
    - 2.1.6 [UT:](#SFG-UT:)
  + 2.2 [Optimum overlap with compute and collective](#SFG-Optimumoverlapwithcomputeandcollect)
    - 2.2.1 [TP with SFG](#SFG-TPwithSFG)
      * 2.2.1.1 [UT description](#SFG-UTdescription)
      * 2.2.1.2 [GC issue with input SRAM reuse](#SFG-GCissuewithinputSRAMreuse)
      * 2.2.1.3 [GC scheduling consideration](#SFG-GCschedulingconsideration)
      * 2.2.1.4 [Control edge](#SFG-Controledge)
      * 2.2.1.5 [GC usability API](#SFG-GCusabilityAPI)
      * 2.2.1.6 [Evaluation for impact with GC scheduler constraint](#SFG-EvaluationforimpactwithGCschedulerc)
      * 2.2.1.7 [Proposed Solution](#SFG-ProposedSolution)
    - 2.2.2 [DDP and SFG](#SFG-DDPandSFG)
  + 2.3 [QA Plan for SFG](#SFG-QAPlanforSFG)
  + 2.4 [SFG and Lazy Updates (suspended)](#SFG-SFGandLazyUpdates(suspended))
    - 2.4.1 [Status: SFG with lazy (suspended)](#SFG-Status:SFGwithlazy(suspended))
    - 2.4.2 [WW14:](#SFG-WW14:)
    - 2.4.3 [WW15:](#SFG-WW15:)
    - 2.4.4 [WW16:](#SFG-WW16:)
    - 2.4.5 [ww17:](#SFG-ww17:)
    - 2.4.6 [ww18:](#SFG-ww18:)
    - 2.4.7 [ww19:](#SFG-ww19:)
    - 2.4.8 [ww20:](#SFG-ww20:)
* 3 [Background](#SFG-Background)
  + 3.1 [Motivation for SFG](#SFG-MotivationforSFG)
  + 3.2 [Design](#SFG-Design)
  + 3.3 [Previous Attempts on SFG](#SFG-PreviousAttemptsonSFG)
  + 3.4 [References:](#SFG-References:)

# Introduction

This page tracks the progress of SFG.

# Goal

The goal for sprint 1.16 for SFG:

1. Apply SFG on the prefill/prompt stage for Llama 70B inference -  [SW-178223](https://jira.habana-labs.com/browse/SW-178223) - Getting issue details... STATUS
2. Architecture for SFG and torch.compile - [SW-171190](https://jira.habana-labs.com/browse/SW-171190) - Getting issue details... STATUS

Apart from these, the following stretch goals will be looked at. This will go beyond 1.16, but they would be essential for the adoption of SFG in future, especially to determine the ROI on torch.compile side going forward.

1. Stretch goal - QA plan for SFG to ensure maturity of stack
2. Stretch goal - Guideline for applicability of SFG for a use case

## Llama 70B Prompt Stage SFG with torch.compile

### Status: SFG with torch.compile

SFG with attention split vs no SFG and no attention split

* Model run: LLAMA3.1 70B model
* Device: Gaudi3
* Build: 1.21.0:275
* Trace files: /software/users/magarwal/sfg\_traces/llama\_3.1\_70b/
  + N**:** /software/users/magarwal/sfg\_traces/llama\_3.1\_70b/without\_SFG/trace\_dir/default\_profiling\_3758888.hltv
  + SFG with attention split: /software/users/magarwal/sfg\_traces/llama\_3.1\_70b/with\_SFG/default\_profiling\_3068033.hltv
* Commands used

hl-prof-config -gaudi3 -e off -phase=multi-enq -use-template profile\_api\_with\_nics -o ./trace\_dir -g 1-1000 --host-start-disabled off

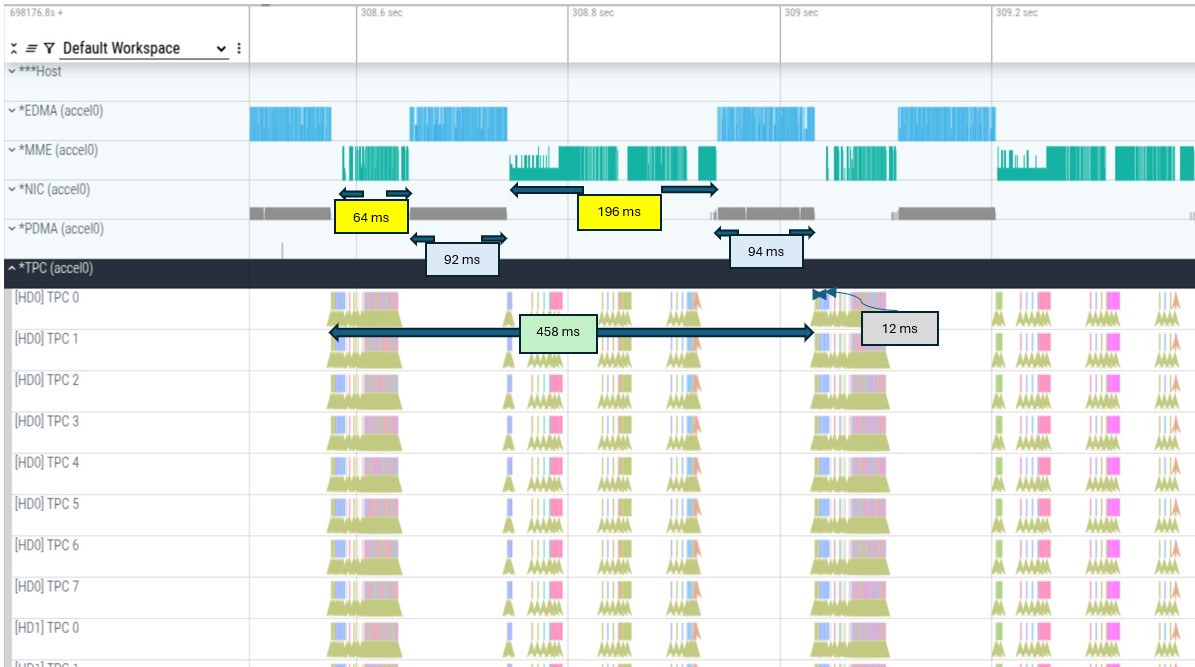
**Without SFG command:**

HABANA\_PROFILE\_WRITE\_HLTV\_WITH\_HOST=1 HABANA\_PROFILE=1 PT\_HPU\_LAZY\_MODE=0 python3 ../gaudi\_spawn.py --use\_deepspeed --world\_size 8 run\_generation.py \  
--model\_name\_or\_path /mnt/weka/data/pytorch/llama3.1/Meta-Llama-3.1-70B-Instruct/ --attn\_softmax\_bf16 \  
--use\_kv\_cache --max\_new\_tokens 2048 --bf16 --batch\_size 330 --use\_flash\_attention --flash\_attention\_recompute \  
--bucket\_size=128 --bucket\_internal --trim\_logits --max\_input\_tokens 2048 --warmup 2 --torch\_compile --regional\_compile --cache\_size\_limit 128

**With SFG command:**

HABANA\_PROFILE\_WRITE\_HLTV\_WITH\_HOST=1 HABANA\_PROFILE=1 PT\_HPU\_ENABLE\_SFG=1 PT\_HPU\_ENABLE\_LAZY\_COLLECTIVES=1 PT\_HPU\_ENABLE\_ALLREDUCE\_GRAPH\_SPLIT=0 PT\_HPU\_LAZY\_MODE=0 PT\_HPU\_ENABLE\_WAITTENSOR\_GRAPH\_SPLIT=0 \  
python3 ../gaudi\_spawn.py --use\_deepspeed --world\_size 8 run\_generation.py   \  
--model\_name\_or\_path /mnt/weka/data/pytorch/llama3.1/Meta-Llama-3.1-70B-Instruct/  --attn\_softmax\_bf16 \  
--use\_kv\_cache --max\_new\_tokens 2048 --bf16 --batch\_size 330 --use\_flash\_attention --flash\_attention\_recompute  \  
--bucket\_size=128 --bucket\_internal  --trim\_logits --max\_input\_tokens 2048  --warmup 2 --torch\_compile --attn\_batch\_split 3  --regional\_compile --cache\_size\_limit 128

* Trace results
  + No SFG and no attention split



* + SFG with attention split



* Throughput measurement

|  |  |  |
| --- | --- | --- |
| **Configurations** | **Throughput** | **Percentage vs Default** |
| Default model run | 6531.15 | 100.00% |
| Model with attention batch split | 6685.38 | 102.36% |
| Model with attention batch split and disabling wait tensor split without SFG | 6628.69 | 101.49% |
| Model with attention batch split and disabling wait tensor split with SFG | 6723.61 | 102.95% |

* Next steps:
  + Evaluate why the prompt stage perf gain doesn't contribute a lot to the overall throughput

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Task** | **JIRA** | **Owner** | **Status** | **Expected date** | **Comments** |
| LLama inference script modification to enable SFG with torch.compile (prompt stage) | [SW-214794](https://jira.habana-labs.com/browse/SW-214794) - Getting issue details... STATUS | Chen,  Amit C (for analysis and bridge support) | Created |  | Enabling SFG with torch.compile on LLama  [SFGLlamaOverlapChanges.pptm](file:///C:\download\attachments\100926802\SFGLlamaOverlapChanges.pptm%3fversion=1&modificationDate=1738057859407&api=v2)  PR: <https://github.com/habana-internal/optimum-habana-fork/pull/98>  PR 2 (tor torch.compile OOM): <https://github.com/habana-internal/optimum-habana-fork/pull/90> |
| Propose user friendly SFG adoption model - explore the decorator approach | [SW-203457](https://jira.habana-labs.com/browse/SW-203457) - Getting issue details... STATUS  [SW-207947](https://jira.habana-labs.com/browse/SW-207947) - Getting issue details... STATUS  [SW-207948](https://jira.habana-labs.com/browse/SW-207948) - Getting issue details... STATUS  [SW-180476](https://jira.habana-labs.com/browse/SW-180476) - Getting issue details... STATUS | Amit, Siva | Open |  | This ticket is to explore more composable design for user adoption for SFG. Some of the things to consider -   * How does user provide a hint about SFG enabling a layer? * Can SFG be enabled as an FX pass for a TP enabled layer? * SFG enable/disable through python API instead of flag   Currently, the model script is being modified manually (for attention split with lazy mode) and torch.compile will start with the required script change. Once the torch.compile SFG starts working on the model, decorator approach can be used.  [Example decorator in UT](https://gerrit.habana-labs.com/#/c/461672/1/tests/pytest_working/compile/test_llamamlpsfg.py) |
| ~~[torch.compile] Error running Llama model with PT\_HPU\_ENABLE\_ALLREDUCE\_GRAPH\_SPLIT=0~~ | [~~SW-212708~~](https://jira.habana-labs.com/browse/SW-212708) ~~- Getting issue details... STATUS~~ | ~~Michal S.~~ | ~~Assigned~~ |  |  |
| ~~[G3][Inference] Hide NIC latency prompt~~ | [~~SW-207965~~](https://jira.habana-labs.com/browse/SW-207965) ~~- Getting issue details... STATUS~~ | ~~Kalyan~~ | ~~Merged~~ |  | [~~https://github.com/habana-internal/optimum-habana-fork/pull/14~~](https://github.com/habana-internal/optimum-habana-fork/pull/14) ~~- attention batch splitting in lazy mode.~~  ~~Note: This doesn't actually use SFG during recipe creation/execution. The work here is to split the batch dim for attention and MLP layers and break the graphs and overlap compute/collective.~~ |
| ~~Need perf improvement and new maxBS info for latency and TPS scenarios below~~ | [~~SW-211161~~](https://jira.habana-labs.com/browse/SW-211161) ~~- Getting issue details... STATUS~~ |  | ~~Done~~ |  | [~~Throughput improvement~~](https://jira.habana-labs.com/browse/SW-211161?focusedId=942882&page=com.atlassian.jira.plugin.system.issuetabpanels%3Acomment-tabpanel#comment-942882) ~~seen, but there is degrade with latency which is to be analyzed~~ |
| [Lazy] memory footprint is higher with attention+mlp split | [SW-212459](https://jira.habana-labs.com/browse/SW-212459) - Getting issue details... STATUS | Kalyan | Review |  |  |
| ~~Gaudi3: Llama2 7b hf bf16 Issue with hpu\_graphs when returning torch.size with no tensor memory~~ | [~~SW-212466~~](https://jira.habana-labs.com/browse/SW-212466) ~~- Getting issue details... STATUS~~ | ~~Kalyan~~ |  |  | ~~disable\_tensor\_cache proposed as a workaround.~~ |
| ~~Modify UT to run with torch.compile~~ | [~~SW-186027~~](https://jira.habana-labs.com/browse/SW-186027) ~~- Getting issue details... STATUS~~ | ~~Siva/Sujoy~~ | ~~Done~~ | ~~WW29~~ | ~~UT added at pytorch-integration/tests/distributed/test\_sfg.py~~ |
| ~~torch.compile enabling with SFG~~ | [~~SW-186887~~](https://jira.habana-labs.com/browse/SW-186887) ~~- Getting issue details... STATUS~~  [~~SW-171190~~](https://jira.habana-labs.com/browse/SW-171190) ~~- Getting issue details... STATUS~~ | ~~Siva~~ | ~~Done~~ | ~~WW29~~ | **~~WW29:~~**   * ~~Patch merged~~   **~~WW28:~~**   * ~~Patch in review. SFG marking passes is enabled with this patch from FX to JIT and enabled end to end SFG flow on torch.compile~~   **~~WW27:~~**   * ~~FX creates functional graphs, this creates multiple slice and slice\_convert. The mm operation is out of place and produces additional tensor - WIP for converting the mm to mm inplace or mm.out variation.~~ * ~~The out-of-place mm output goes to allreduce. The graph and UT has been modified to ensure that the mm are all in same graph followed by allreduce. This allows single recipe creation with all mm nodes. SFG marking is working with the WIP patch.~~ * ~~allreduce->wait\_tensor output is copies back to the original output with slice\_scatter, before the output is given to add op. This is resulting in a copy before the add. Moving the mm to inplace/mm.out is expected to resolve this~~ |
| ~~in-place collective in FX graph~~ | [~~SW-187076~~](https://jira.habana-labs.com/browse/SW-187076) ~~- Getting issue details... STATUS~~ | ~~Siva~~ | ~~Done~~ | ~~WW25~~ |  |
| ~~Avoid collective reordering with partitioner~~ | [~~SW-188215~~](https://jira.habana-labs.com/browse/SW-188215) ~~- Getting issue details... STATUS~~ | ~~Siva~~ |  |  | ~~Status:~~  **~~WW30:~~**   * ~~No impact on SFG - Siva to update the Jira and close.~~ |
| ~~Memory overhead reduction with SFG and torch.compile~~ | [~~SW-191423~~](https://jira.habana-labs.com/browse/SW-191423) ~~- Getting issue details... STATUS~~ | ~~Siva~~ | ~~Analysis done, close?~~ | ~~WW31~~ | ~~Status:~~  **~~WW30:~~**   * ~~Experiment with making the slice nodes logical and engage GC and bridge folks~~   **~~WW29:~~**   * ~~Measurement done on UT and SFG doesn't add performance degradation.~~ * ~~There is follow-up work to reduce the copy operations, which doesn't add to memory but is required for performance benefit.~~   **~~WW27:~~**   * ~~This should be resolved with the plan for~~ [~~SW-186887~~](https://jira.habana-labs.com/browse/SW-186887) ~~- Getting issue details... STATUS , but the memory needs to be measured to ensure there is no overhead~~ |
| ~~Add minimum latency node attribute on Synapse graph for SFG external tensors~~ | [~~SW-187919~~](https://jira.habana-labs.com/browse/SW-187919) ~~- Getting issue details... STATUS~~ | ~~Amit C~~ | ~~Review~~ |  | ~~Dependent on~~ [~~SW-188353~~](https://jira.habana-labs.com/browse/SW-188353) ~~- Getting issue details... STATUS~~  **~~WW30:~~**   * ~~Patch review ongoing, will be added once Synapse API patch is merged.~~   **~~WW29:~~**   * ~~Patch under review and test~~ |
| ~~Re-inplace SFG producing op to avoid copy or slice\_scatter~~ | [~~SW-192995~~](https://jira.habana-labs.com/browse/SW-192995) ~~- Getting issue details... STATUS~~ | ~~Siva~~ | ~~Assigned~~ |  | **~~WW29:~~**   * ~~Analysis started for avoiding the copy~~ * ~~In order to remove the copy, the mm op needs to be modified to become mm.out variant and produce output in the provided output shard~~ * ~~Needs the mm.out to work on the ops side for torch.compile~~ * ~~Two issues raised~~ [~~SW-193237~~](https://jira.habana-labs.com/browse/SW-193237) ~~- Getting issue details... STATUS and~~ [~~SW-193241~~](https://jira.habana-labs.com/browse/SW-193241) ~~- Getting issue details... STATUS~~ |
| ~~[PT][t.compile] Wrong dependency graph created by mm.out op for GC~~ | [~~SW-193237~~](https://jira.habana-labs.com/browse/SW-193237) ~~- Getting issue details... STATUS~~ | ~~Ramesh~~ | ~~Closed~~ |  |  |
| ~~[PT][t.compile] Persistent tensor information not filled correctly in case of non-functional graphs~~ | [~~SW-193241~~](https://jira.habana-labs.com/browse/SW-193241) ~~- Getting issue details... STATUS~~ | ~~Marcin~~ | ~~Closed~~ |  |  |
| Minimize node latency | [SW-188353](https://jira.habana-labs.com/browse/SW-188353) - Getting issue details... STATUS | Shay | In Progress |  | Status:  **WW27:**   * ETA: Support in 1.18. I hope to have the API before 1.17 feature freeze |
| ~~Enable SFG Mechanism in Gaudi3~~ | [~~SW-203485~~](https://jira.habana-labs.com/browse/SW-203485) ~~- Getting issue details... STATUS~~ | ~~Shay~~ |  |  | ~~Done~~ |
| ~~Gaudi3: mark tensor as external fails in Gaudi3~~ | [~~SW-203485~~](https://jira.habana-labs.com/browse/SW-203485) ~~- Getting issue details... STATUS~~ | ~~Shay~~ |  |  | ~~Done~~ |
| ~~Modify UT to include the MLP layer from Llama~~ | [~~SW-203449~~](https://jira.habana-labs.com/browse/SW-203449) ~~- Getting issue details... STATUS~~ | ~~Siva~~ | ~~WIP~~ |  | ~~Update the UT to work on the MLP layer from Llama.~~ |
| ~~Extend SFG in UT to more than the final matmul from MLP layer~~ | [~~SW-203450~~](https://jira.habana-labs.com/browse/SW-203450) ~~- Getting issue details... STATUS~~ | ~~Siva~~ | ~~WIP~~ |  | ~~Enable SFG on the entire MLP layer with batch dim sharding across the ops -~~  ~~self.w2(F.silu(self.w1(x)) \* self.w3(x)), w1 and w3 are column parallel linear and w2 is the row parallel linear with TP.~~ |
| Overlap the compute by breaking graph with graph break |  | Sujoy |  |  | Gaudi3: Possibly an overlap solution on lazy, with enough compute on the shards. |
| Memory increase with torch.compile | [SW-191762](https://jira.habana-labs.com/browse/SW-191762) - Getting issue details... STATUS | Mansi |  |  | Run with the patches that plans to reduce the memory, so that we can use the BS=370. |
| SFG QA | [SW-194208](https://jira.habana-labs.com/browse/SW-194208) - Getting issue details... STATUS | Swadesh/Mansi | Open |  | [QA plan provided and reviewed](https://confluence.habana-labs.com/display/Frameworks/SFG#SFG-QAPlanforSFG)  **WW30:**   * Swadesh/Mansi to add QA functional tests for SFG |
| large hcclAllReduce time blocking performance for LLama prompt stage SFG | [SW-181667](https://jira.habana-labs.com/browse/SW-181667) - Getting issue details... STATUS | Mansi | Open | No plan | Status: Measure this once the SFG feature is enabled  **WW27:**   * Once [SW-186887](https://jira.habana-labs.com/browse/SW-186887) - Getting issue details... STATUS is resolved, ensure the collective times are not increasing compared to non-SFG case.   **WW25:**   * The time with 8 cards are within the theorical limit. No action item for this now with the 8 TP configuration. However, it is an open if the TP config is reduced. |

### Llama config for SFG:

BS=370, input=2048, output=2048, TP=8, precision=BF16

### Prompt stage tensor shapes:

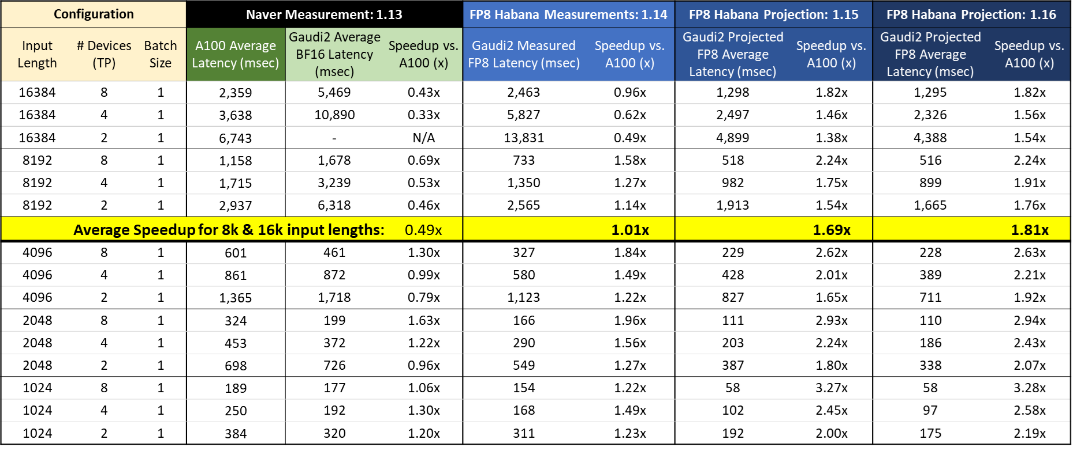
The LinearAllReduce layer configurations:

ScopedLinearAllReduce:: Shape of input = torch.Size([370, 2048, 1024])  
ScopedLinearAllReduce:: Shape of weight = torch.Size([8192, 1024])  
ScopedLinearAllReduce:: Shape of self.weight.transpose(-1, -2).shape = torch.Size([1024, 8192])

ScopedLinearAllReduce:: Shape of input = torch.Size([370, 2048, 3584])  
ScopedLinearAllReduce:: Shape of weight = torch.Size([8192, 3584])  
ScopedLinearAllReduce:: Shape of self.weight.transpose(-1, -2).shape = torch.Size([3584, 8192])

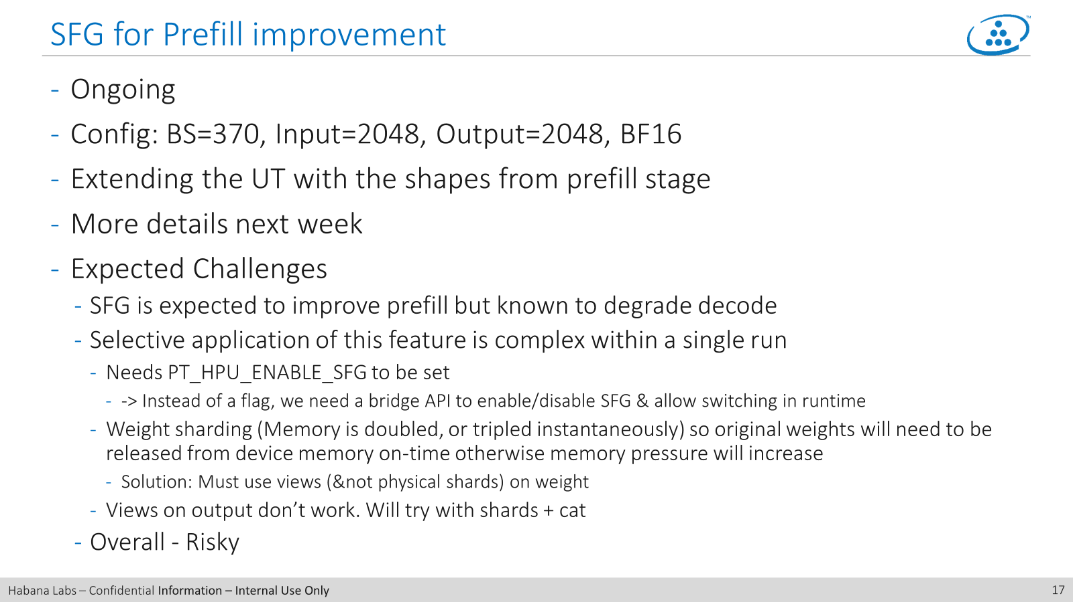
### Projection:

The projection from Roman for SFG on LLama:



The improvement for input length=2048 is marginal for BS=1. Hence, a BS=370 is chosen.

### Design considerations from Deepak:



### UT:

The UT is WIP. An initial version is available at /software/users/ssaraswati/tests/example\_tptest\_llama\_prompt\_sfg\_layer1.py

The test checks creates a single layer with the weight size [8192, 1024] and input size [370, 2048, 1024]. The matmul is done in the following way -

transpose the weight → size [1024, 8192]

For 4 shard of the weight, do the following operation:

   each shard size = 8192 / 4 = 2048

    output\_shard = torch.matmul(input, transposed weight[:,start\_offset : start\_offset + curr\_shard\_size])

    This is equivalent to -

    output\_shard size  [370, 2048, 2048 ] = torch.matmul(input size [370, 2048, 1024], transposed weight shard size [1024, 2048])

     After this, do a all\_reduce of the output\_shard

     Append the output\_shard to a output list

 At the end of the loop, to a torch.cat of the 4 output shard to a single output of size [370, 2048, 8192]

 Add the bias size [8192] to it.

## Optimum overlap with compute and collective

This section discussed the proposal for overlapping the compute and collective optimally with SFG.

### TP with SFG

#### UT description

As described in [SW-182856](https://jira.habana-labs.com/browse/SW-182856) - Getting issue details... STATUS , the UT has been designed to break the matmul (from the linear layer) into multiple shards, and thereby allowing trigger of collective in pipeline with the compute. The transformation of the UT is described below -

**UT one shard code**

class FullLinear(nn.Module):

def \_\_init\_\_(self, weight\_size, dtype):

super().\_\_init\_\_()

# with Tx, self.weight is = 1024, 8192

self.weight = nn.Parameter(torch.arange(weight\_size[1] \* weight\_size[0], dtype = dtype).reshape(weight\_size[1], weight\_size[0]))

# bias size = 8192

self.bias = nn.Parameter(torch.empty(1, weight\_size[0], dtype = dtype).fill\_(0.01))

def forward(self, input, dummy\_tensor\_for\_hccl\_sync):

work = torch.distributed.all\_reduce(dummy\_tensor\_for\_hccl\_sync, async\_op=True)

work.wait()

output = torch.matmul(input, self.weight)

torch.distributed.all\_reduce(output, async\_op=True)

return output + self.bias

This has been transformed into shards the following way. The description of the issue is given with lazy + SFG first, to demonstrate the requirement at the bridge side.

**UT with shards for lazy + SFG**

def forward(self, input, dummy\_tensor\_for\_hccl\_sync):

# torch.matmul(input size=(370, 2048, 1024), weight size = (1024, 8192) + bias (size = 8192)

# output size = (370, 2048, 8192)

total\_shard\_size = self.weight.size()[1]

shard\_size = int(total\_shard\_size//self.num\_shards)

start\_offset = 0

works = []

output\_shards = []

# This is a dummy collective to bridge all cards in sync before the actual loop below. Not expected in models.

work = torch.distributed.all\_reduce(dummy\_tensor\_for\_hccl\_sync, async\_op=True)

work.wait()

for i in range(self.num\_shards):

curr\_shard\_size = shard\_size if i < self.num\_shards-1 else total\_shard\_size - start\_offset

output\_shard = torch.matmul(input, self.weight[:,start\_offset : start\_offset + curr\_shard\_size])

work = torch.distributed.all\_reduce(output\_shard, async\_op=True)

works.append(work)

output\_shards.append(output\_shard)

start\_offset = start\_offset + curr\_shard\_size

output = torch.cat(output\_shards, dim = 2)

output\_shards = []

return output + self.bias

The corresponding UT with torch.compile + SFG is WIP : [SW-186027](https://jira.habana-labs.com/browse/SW-186027) - Getting issue details... STATUS , with enabling patches for torch.compile + SFG [SW-186887](https://jira.habana-labs.com/browse/SW-186887) - Getting issue details... STATUS and [SW-187076](https://jira.habana-labs.com/browse/SW-187076) - Getting issue details... STATUS . Once the torch.compile SFG is merged, the UT and corresponding evaluation of compute and collective overlap will be switched to it. However, from the point of view of the recipe, the description here is expected to be same for lazy as well as torch.compile.

**UT with shards for torch.compile + SFG**

def forward(self, inp, dummy\_tensor\_for\_hccl\_sync):

# torch.matmul(input size=(370, 2048, 1024), weight size = (1024, 8192) + bias (size = 8192)

# output size = (370, 2048, 8192)

total\_shard\_size = inp.size()[0]

shard\_size = int(total\_shard\_size//self.num\_shards)

start\_offset = 0

output = torch.empty([inp.size()[0], inp.size()[1], self.weight.size()[1]], dtype = inp.dtype, device="hpu")

       # This is a dummy collective to bridge all cards in sync before the actual loop below. Not expected in models.

        work = torch.ops.c10d\_functional.all\_reduce(dummy\_tensor\_for\_hccl\_sync, "sum", "default", [0, 1, 2, 3], 4)

wait\_t = torch.ops.c10d\_functional.wait\_tensor(work)

for i in range(self.num\_shards):

curr\_shard\_size = shard\_size if i < self.num\_shards-1 else total\_shard\_size - start\_offset

# Slice on the batch dim of input and output

output\_shard = output[start\_offset : start\_offset + curr\_shard\_size, :, :]

torch.matmul(inp[start\_offset : start\_offset + curr\_shard\_size, :, :], self.weight, out = output\_shard)

work = torch.ops.c10d\_functional.all\_reduce(output\_shard, "sum", "default", [0, 1, 2, 3], 4)

output\_shard = torch.ops.c10d\_functional.wait\_tensor(work)

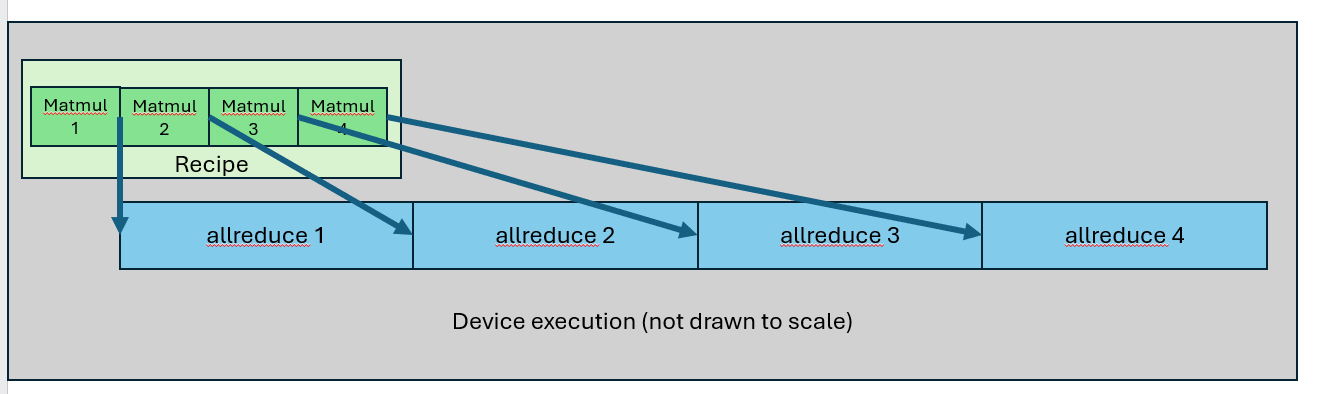
start\_offset = start\_offset + curr\_shard\_size

return output + self.bias

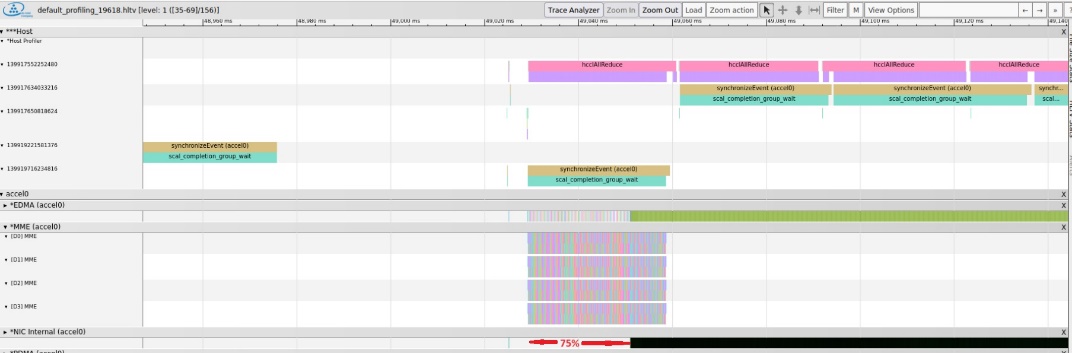
#### GC issue with input SRAM reuse

From [SW-182856](https://jira.habana-labs.com/browse/SW-182856) - Getting issue details... STATUS -

the expectation is to run the UT in the following way. If there are 4 shards, then there are 4 matmul and the allreduce should get triggered at the end of each matmul. The picture below shows the ideal expected overlap in the device, where each matmul triggers the corresponding collective. For 4 shards, this means after 25% (1st of the 4 matmuls) of the compute recipe execution, collective will start executing in device.

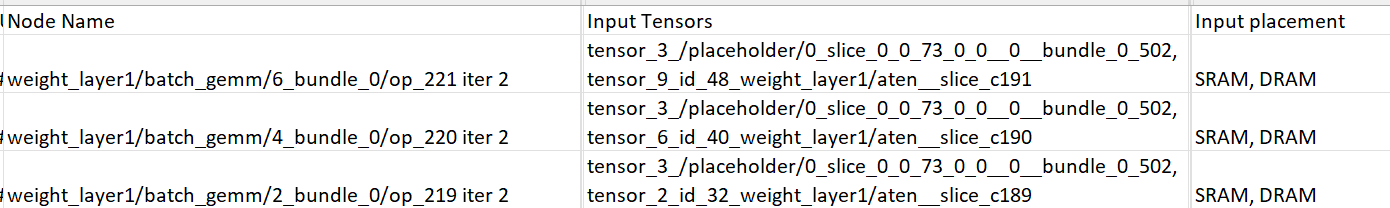


However, from /software/users/ssaraswati/tests/SFG\_prompt\_Llama\_70B\_inference/final\_logs/BS370/4shards/default\_profiling\_19618.hltv, it is evident that the input slices are kept in SRAM by GC, and is shared for matmul with the weight slices. The input slices from SRAM are shared for the first three weight slices, hence the first, second and third matmul also ends together, at around 75% of the MME execution. The 4th matmul is done after this. This can be seen from the trace snapshot below -

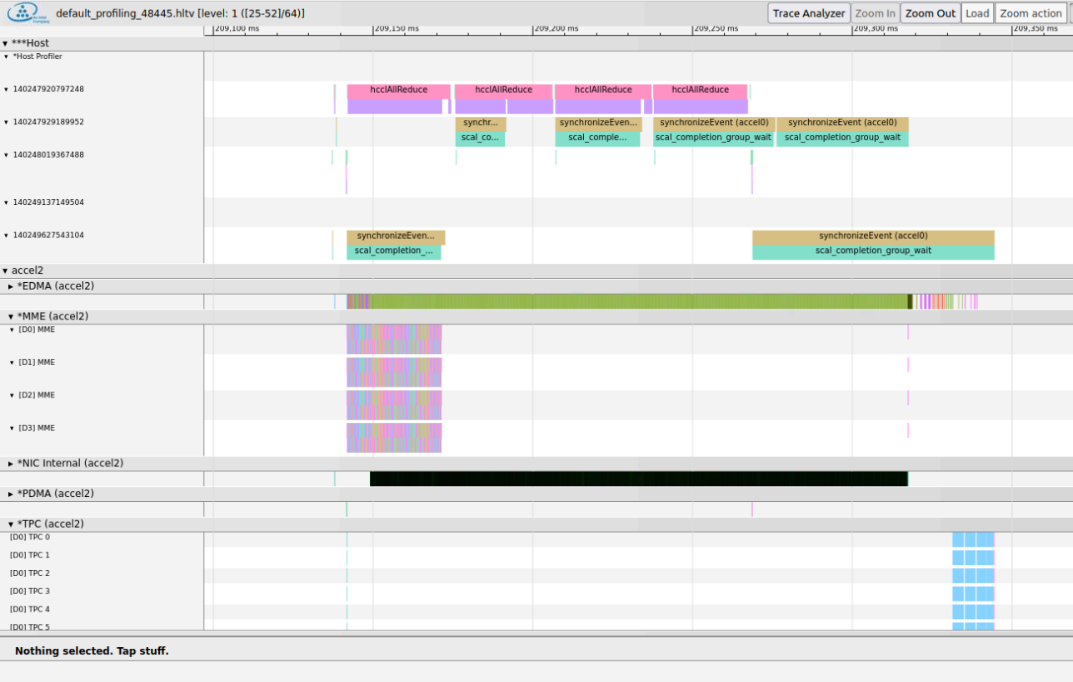


The reason behind this issue is that the GC puts the slices from first 3 matmul in the same bundle. The input is shared by all matmuls, hence GC decides to optimize the SRAM use and keeps an input slice in SRAM, and brings weight slices for the first 3 matmuls from HBM to multiple with the input slice in SRAM. This can be seen from the traceanalyzer snapshot below. The input slice "tensor\_3\_/placeholder/0\_slice\_0\_0\_73\_0\_0\_\_0\_\_bundle\_0\_502" is in SRAM. In the same bundle, three gemm ops are there using this input SRAM slice and multiplying with the weight bundles from 3 different matmuls.

This essentially means first 3 matmuls are running together in bundle 0 and finishing together after 75% of the compute. Only after this, the SFG signals for all 3 matmuls are triggered which results in the trace above.



The workaround with [SW-183656](https://jira.habana-labs.com/browse/SW-183656) - Getting issue details... STATUS , with SRAM\_SLICER\_SHARED\_MME\_INPUT\_EXPANSION\_ENABLED=false shows that if there is no SRAM slice sharing, then the collectives are triggered after each matmul -



#### GC scheduling consideration

As per the [comment](https://jira.habana-labs.com/browse/SW-182856?focusedId=767135&page=com.atlassian.jira.plugin.system.issuetabpanels:comment-tabpanel#comment-767135) in [SW-182856](https://jira.habana-labs.com/browse/SW-182856) - Getting issue details... STATUS -

SRAM\_SLICER\_SHARED\_MME\_INPUT\_EXPANSION\_ENABLED=false is the hacky W/A. It will degrade performance of, for example, bwd layers even if there is only a single SFG in these layers.

As a general solution, we would like to add some constrains on the number of SFG tensor producers that can be bundled together.  
For example, a naive constraint would be to limit to 1 producer.  
A more advanced constraint can also make sure not to bundle anything that increases the latency of an SFG producer (for example not bundling with a consumer), but this may have too large effect on the single device time, so it will need careful consideration.

#### Control edge

To constrain the GC scheduler bundling on SFG output producers, there are some choices described above. This could be done in two places -

1. GC: The synapse graph is already marked with out tensors that are marked with external SFG signal. Hence, GC could look into the graph and decide the scheduling policy on how many producers should be bundled together.
2. Bridge: Bridge can add a control edge from one SFG producer to another, which will constraint the GC not to bundle them together.

The second option was chosen and [SW-186029](https://jira.habana-labs.com/browse/SW-186029) - Getting issue details... STATUS is planned in bridge to add the control edges.

#### GC usability API

GC is adding a [user programmability API](https://gerrit.habana-labs.com/#/c/411396/5/include/synapse_api.h) (example test [here](https://gerrit.habana-labs.com/#/c/411396/5/tests/gc_tests/unit_tests/gaudi2/programmability_tests.cpp)). This can be used to also explicitly set the ordering index for nodes in a Synapse graph and is an alternate implementation choice to control edges. Bridge will evaluate this along with control edge direction after aligning with GC plan for adding this interface.

Along with the GC API, a Higher order operation can be enabled for users to provide hints. This can be an additional tool for advanced users who wants to control the compute bundling with SFG. However, this is not proposed to be added in the beginning. This path will be added only after delivering the TP and torch.compile SFG and after doing the exercise mentioned below on the impact of SFG on compute performance.

#### Evaluation for impact with GC scheduler constraint

One downside of limiting the GC scheduler bundling for SFG tensor producers is a compromise on the recipe optimization. The TP scenario with Llama is heavily biased towards low compute and high collective execution time, hence marginal increase in compute execution is beneficial if the collective can be overlapped as early as possible. However, the general impact on scenarios like DDP and SFG is not known. The plan for addressing this stated below -

* Bridge to add the control edges for all SFG marked tensors
* Evaluate the UT performance and achieve desired compute and collective overlap for LLama TP scenario
* Enable this solution for workload by 1.18
* Enable DDP and torch.compile SFG with the same control edge added from bridge
* Evaluate the DDP with SFG performance impact. If the performance is seen to be improving, then enable this on workloads.

#### Proposed Solution

The proposed solution is [SW-188353](https://jira.habana-labs.com/browse/SW-188353) - Getting issue details... STATUS , the description from GC -

* Signaling for graph is used to create pipeline over NIC. Nodes that produce SFG tensor latency should be minimized, otherwise the pipeline over NIC is not optimal or not achieved. For example: if two nodes that produce SFG tensors are bundled the tensors will be signaled  closely and NIC operation won't have good pipeline.

Synapse API requirements [SW-191632](https://jira.habana-labs.com/browse/SW-191632) - Getting issue details... STATUS

* + Synapse will expose an optional attribute per node ( default: false) to indicate that GC should try to minimize node latency. The API proposed is synNodeSetMinimalLatency()

GC Design Requirements

* + Avoid adding a node to bundle that will delay a node that is marked  to minimize latency (i.e. direct/indirect consumer for node that is marked "minimal latency")
  + Schedule earlier nodes marked with "minimal latency"

Bridge Requirements [SW-186887](https://jira.habana-labs.com/browse/SW-186887) - Getting issue details... STATUS , [SW-187919](https://jira.habana-labs.com/browse/SW-187919) - Getting issue details... STATUS

* + Bridge will run FX passes to identify collective producers that are candidate for SFG. These nodes metadata will be marked with additional "sfg" flag.
  + The "sfg" metadata will be propagated via JIT graph to bridge Synapse lowering side. Bridge uses this metadata to mark Syanpse graph output tensors with "external" marking.
  + For the tensors that are marked "external", bridge will use synNodeSetMinimalLatency() to mark the "minimal latency" attribute on the producing Synapse node of the "external" marked tensor

### DDP and SFG

While the 1.18 business ask is to deliver SFG with torch.compile on LLama inference TP with [SW-178223](https://jira.habana-labs.com/browse/SW-178223) - Getting issue details... STATUS , SFG as an infrastructure is also applicable with DDP. Since SFG with lazy didn't show any performance benefit, this is not a business committed goal yet. However, the solution will enable this and workload performance will be evaluated with torch.compile + SFG to decide whether this will be enabled or not.

* Collective fusion is required for DDP
* However, for TP and SFG - the collective fusion should not be triggered
  + It can be currently disabled with PT\_HPU\_DISABLE\_fuse\_allreduce\_calls
  + Inductor has a config.\_fuse\_ddp\_communication flag to enable/disable the collective fusion. <https://gerrit.habana-labs.com/#/c/440078/> patch disables HPU collective fusion with same config.
  + Other than this, the collective fusion is done with DDP buckets. The default bucket (when DDP is not used) \_fuse\_ddp\_bucket\_size is 25 MB (for TP use cases, the size may prevent a fusion of collectives but this isn't a robust way to prevent collective fusion to happen with TP and SFG
* SFG can also be enabled with DDP
  + Run the SFG pass after bucket coalescing pass
* in-place collectives
  + Once the collectives are decided (coalesced or separate), an in-place conversion pass based on pattern match will be run before SFG marking pass. This pattern match is expected not to actually change the DDP bucketed collectives, but can be taken up separately if memory can be saved with it
  + Pass order:
    - bucket coalescing: protected with a flag
    - in-place of collectives: pattern match based
    - SFG pass: when enabled with SFG flag

## QA Plan for SFG

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task** | **JIRA** | **Owner** | **Status** | **Comments** |
| Create a test plan for SFG maturity |  | Swadesh/Mansi | Review | The idea is to create tests that would -   * Identify usual compute+collective scenarios used in workloads, distributed (DDP/Deepspeed), TP/PP etc. * Create compute layer scenarios (with different shape, views, dtypes etc.) * Statically determine pass criterion as determined by the test scenario -   + Bridge creating right graphs with compute, collective and external tensors for SFG   + GC generating correct post graph for SFG * Correct test results |
| Create a test plan for SFG performance |  | Swadesh/Mansi | Review | Characterize the performance scenarios where SFG can be profitable to use -   * Ideal compute and collective mix * Tensor attributes like shape and dtypes * Compute and collective overlap, no. of cards   This is a more difficult task as the signal generation need to be checked based on device traces and logs and this could be a performance characterization task driven with manual intervention initially, which can be later used for performance regression test suite. |
| Test plan document: [SFG\_Validation\_TestPlan.docx](file:///C:\download\attachments\100926802\SFG_Validation_TestPlan.docx%3fversion=1&modificationDate=1721312651197&api=v2) | | | | |

## SFG and Lazy Updates (suspended)

### Status: SFG with lazy (suspended)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Task** | **JIRA** | **Owner** | **Status** | **Expected date** | **Comments** |
| Baseline |  | Deepak | WIP | WW13 | Provide the baseline numbers and traces before SFG is applied |
| Prompt stage tensor data Llama config for SFG |  | Deepak | Done | ~~WW13~~ | Provide the various tensor dimensions for the prompt stage. |
| Projection |  | Deepak/Sujoy | WIP | ~~WW14~~ | Provide a projection of estimated gain based on prefill/prompt stage  Included the projection from Roman. |
| ~~PoC (on a unit test)~~ | [SW-180510](https://jira.habana-labs.com/browse/SW-180510) - Getting issue details... STATUS | Sujoy | Done | ~~WW13~~ | Updated the PoC from previous decode phase exercise to the prompt stage tensor shapes  Verified the UT to see if the shapes, shards, ops and distributed work are as expected  UT is run to ensure correctness - Mansi from Swadesh's team  help in running UT and gathering profile data |
| ~~Analysis of PoC against projection~~ |  |  |  |  | Bridge: Amit to support  Gather the trace without SFG to get baseline for UT.  Create projection from the UT  Enable SFG and gather trace  Compare against projection |
| ~~Graph check : JIT IR~~ | [SW-181338](https://jira.habana-labs.com/browse/SW-181338) - Getting issue details... STATUS | Sujoy | Done | ~~WW14~~ | JIT graphs shows the matmul and allreduce are captured in a single graph |
| ~~Graph check: Synapse pre and post IR~~ | [SW-181338](https://jira.habana-labs.com/browse/SW-181338) - Getting issue details... STATUS | Sujoy | Done | ~~WW14~~ | Synapse graphs:  Post graph for the matmul:  Each matmul is split into 74 slices. The picture shows some details from one of the concat cards. The Batchgemm pre-graph output size is [370, 2048, 2048] which is split into 74 tensors of size [5, 2048, 2048] |
| ~~End to end time: 1shard vs 4shards~~ |  | Mansi | Done | ~~WW15~~ |  |
| ~~Host trace analysis~~ |  | Mansi | Done | ~~WW14~~ |  |
| ~~Device trace analysis~~ |  | Mansi, Sujoy,Deepak | Done | ~~WW14~~ | Raised issue on hcclAllReduce |
| large hcclAllReduce time blocking performance for LLama prompt stage SFG | [SW-181667](https://jira.habana-labs.com/browse/SW-181667) - Getting issue details... STATUS | Adam |  |  |  |
| ~~Trace collection issue for both host and NIC device traces~~ | [SW-181974](https://jira.habana-labs.com/browse/SW-181974) - Getting issue details... STATUS | Mansi | Done | ~~WW16~~ | Issue with trace collection capturing host, NIC, TPC and MME together |
| GC should not reuse slices from SRAM across ops that produce output marked for SFG | [SW-182856](https://jira.habana-labs.com/browse/SW-182856) - Getting issue details... STATUS |  |  |  | MME and NIC overlap is non-optimal - GC is reusing input slices for first 3 matmul, and the SFG trigger for first matmul only comes at around 75% of MME |
| ~~Remove strided insert from collective~~ | [SW-165152](https://jira.habana-labs.com/browse/SW-165152) - Getting issue details... STATUS | Amit C. | Not required, lazy specific |  | To ensure graphs don't break on collective when output is slice |
| ~~Avoid the output copies and cat by updating matmul on output offsets~~ | [SW-183293](https://jira.habana-labs.com/browse/SW-183293) - Getting issue details... STATUS | Amit C | Not required, lazy specific |  |  |

### WW14:

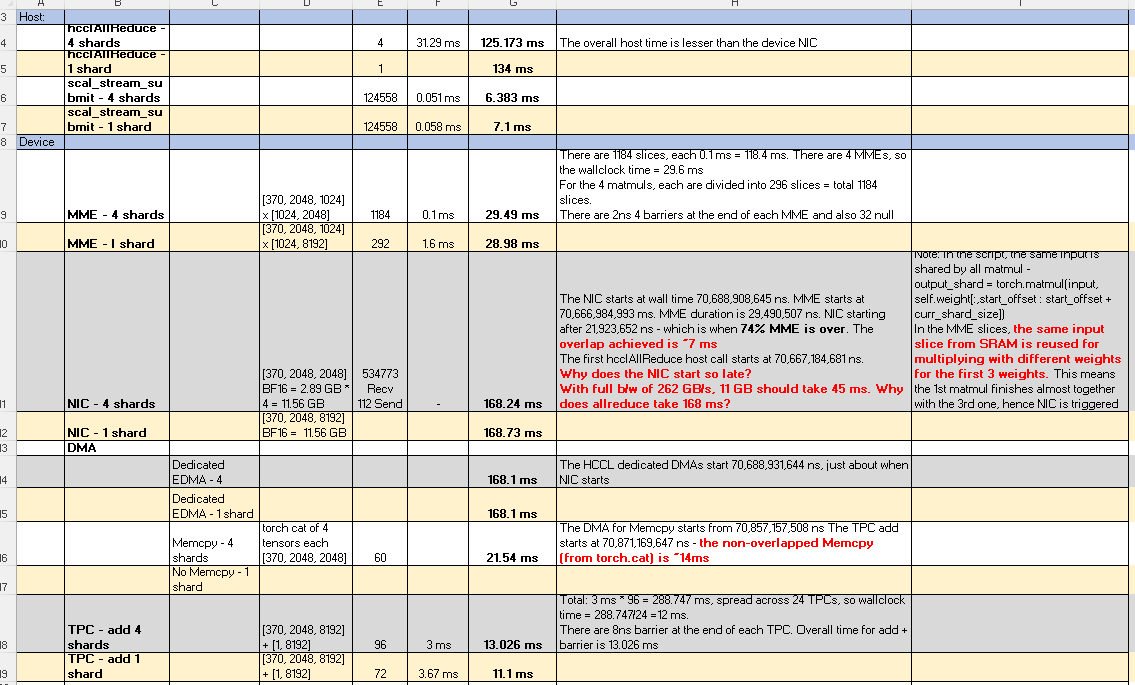
* Captured traces for BS 370, there are issues in getting NIC, MME, TPC and host in same trace
* From analysis of the trace, the hccl calls on host seem to get blocked and device NIC starts after a long time - in discussion with Adam

### WW15:

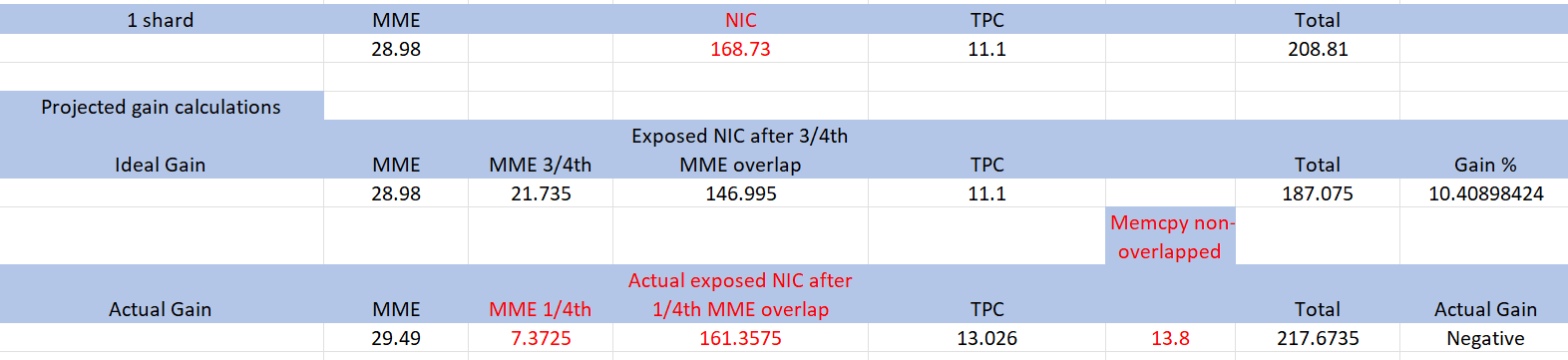
* Trace captured with BS 16, showing all device and host parts and shows the hccl time issue
* Analysis being done on what tensor size for allreduce should be used.
  + Open: What is NIC blocked on? What data do we need to collect apart from the traces?

### WW16:

* Analysis [Projection.xlsx](file:///C:\download\attachments\100926802\Projection.xlsx%3fversion=1&modificationDate=1713343044654&api=v2)



* Projection and issues -



* The main issues
  + allreduce device NIC time (even in 1 shard) is higher
    - Sometimes, the cards are out of sync, further increasing NIC time
  + MME and NIC is not overlapped well - NIC starts only at 75% of MME, reducing the potential benefit
    - GC issue raised. Workaround to allocate copies of input will increase memory pressure. Unit test is getting an DFA
  + Output from matmul is separate tensors which are combined with torch.cat before the bias add. This adds a ~14 ms overhead on DMA between NIC and TPC add
    - Workaround is to try using a output slice that would be produced from matmul. However, this is causing graph breaks in the bridge side.

### ww17:

* GC issue: [SW-182856](https://jira.habana-labs.com/browse/SW-182856) - Getting issue details... STATUS
  + DFA logs seen with the UT with GC environment variables and 4 shards [SW-183656](https://jira.habana-labs.com/browse/SW-183656) - Getting issue details... STATUS

1. SRAM\_SLICER\_SHARED\_MME\_INPUT\_EXPANSION\_ENABLED=false ENABLE\_EXPERIMENTAL\_FLAGS=true

2. PIPELINE\_MANAGEMENT\_FORCE\_BUNDLIZER=1 ENABLE\_EXPERIMENTAL\_FLAGS=true

* + Error seen with higher shards (shard = 16) [SW-183657](https://jira.habana-labs.com/browse/SW-183657) - Getting issue details... STATUS
* Memory log from UT:
  + **1 shards:** in\_use\_memory=13.03 GB, max\_use\_memory=26.04 GB

**4 shards**: in\_use\_memory=24.59 GB, max\_use\_memory=37.6 GB

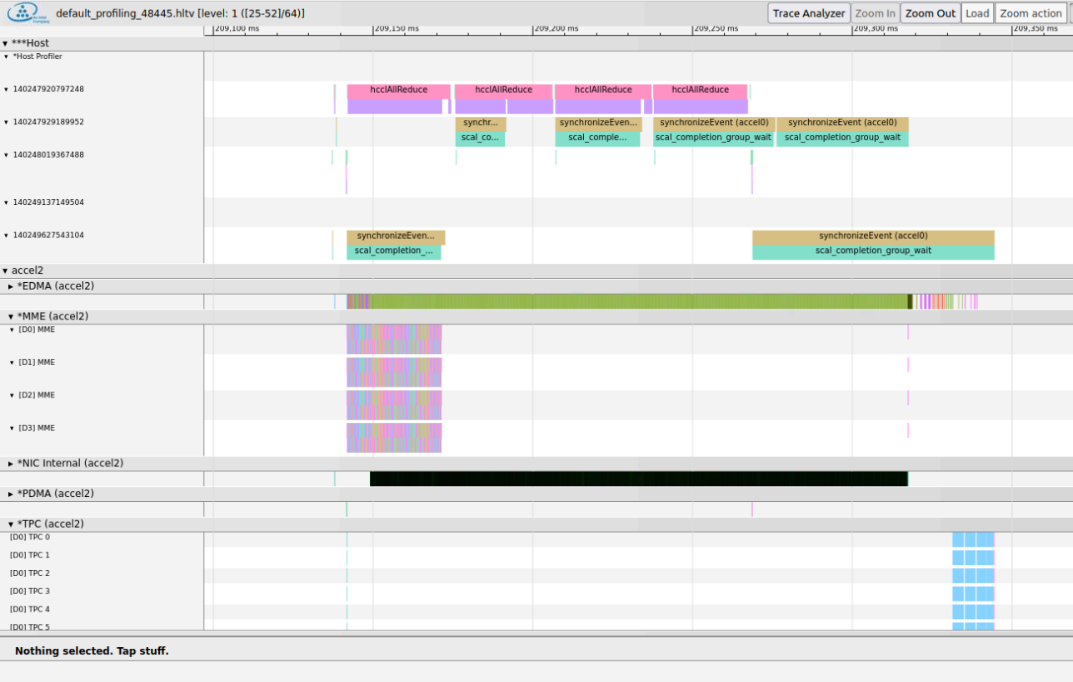
* + Next step: possibly an UT issue, remove the output shards and retry
* Removing torch.cat: [SW-183293](https://jira.habana-labs.com/browse/SW-183293) - Getting issue details... STATUS
  + WIP

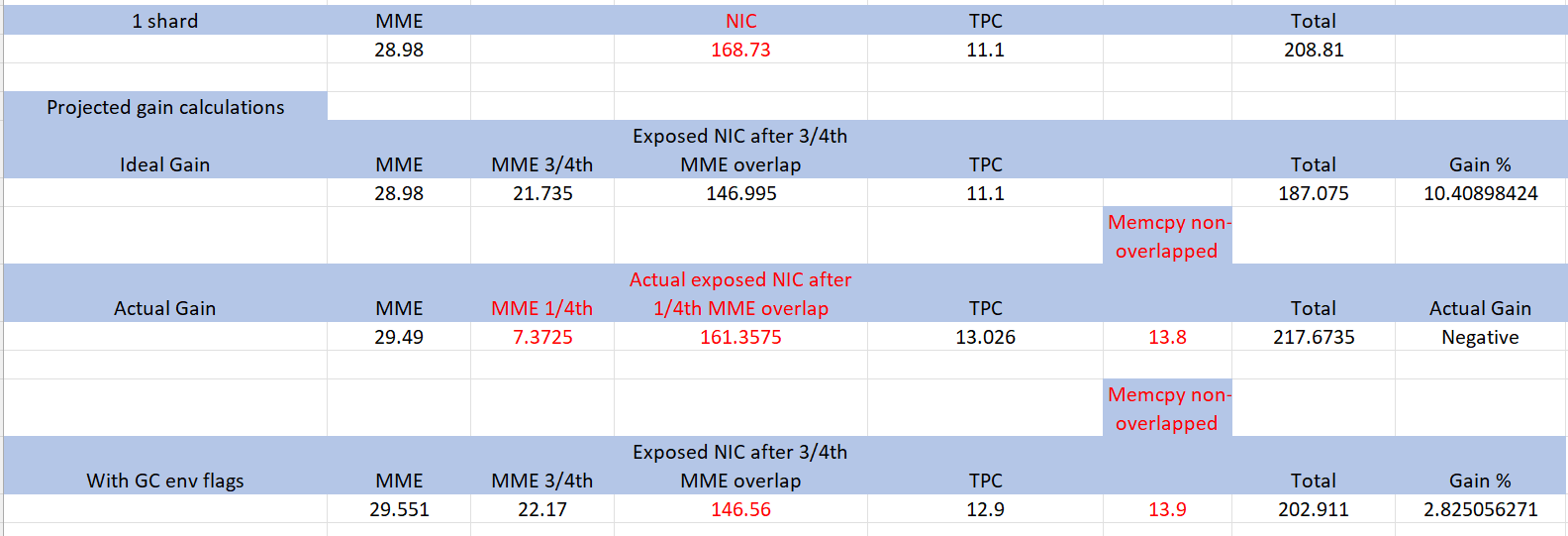
### ww18:

* GC - DFA issue:
  + [SW-183656](https://jira.habana-labs.com/browse/SW-183656) - Getting issue details... STATUS , [SW-183657](https://jira.habana-labs.com/browse/SW-183657) - Getting issue details... STATUS
* Bridge - Removing torch.cat
  + [SW-183293](https://jira.habana-labs.com/browse/SW-183293) - Getting issue details... STATUS
* HCL - high NIC time
  + [SW-181667](https://jira.habana-labs.com/browse/SW-181667) - Getting issue details... STATUS

### ww19:

* Workaround from GC: flags SRAM\_SLICER\_SHARED\_MME\_INPUT\_EXPANSION\_ENABLED=false ENABLE\_EXPERIMENTAL\_FLAGS=true DISABLE\_LOAD\_DIFF\_DESC=1





### ww20:

* GC - DFA issue resolved
  + [SW-183656](https://jira.habana-labs.com/browse/SW-183656) - Getting issue details... STATUS , [SW-183657](https://jira.habana-labs.com/browse/SW-183657) - Getting issue details... STATUS
* Bridge - Removing torch.cat and use control edge for the external signal marked tensors - WIP
  + [SW-183293](https://jira.habana-labs.com/browse/SW-183293) - Getting issue details... STATUS
* HCL - high NIC time
  + [SW-181667](https://jira.habana-labs.com/browse/SW-181667) - Getting issue details... STATUS

# Background

## Motivation for SFG

The motivation for SFG is to start executing collective operations as soon as their respective inputs are ready without waiting for the whole graph to finish execution, thus avoiding graph breaks for collectives to be triggered.

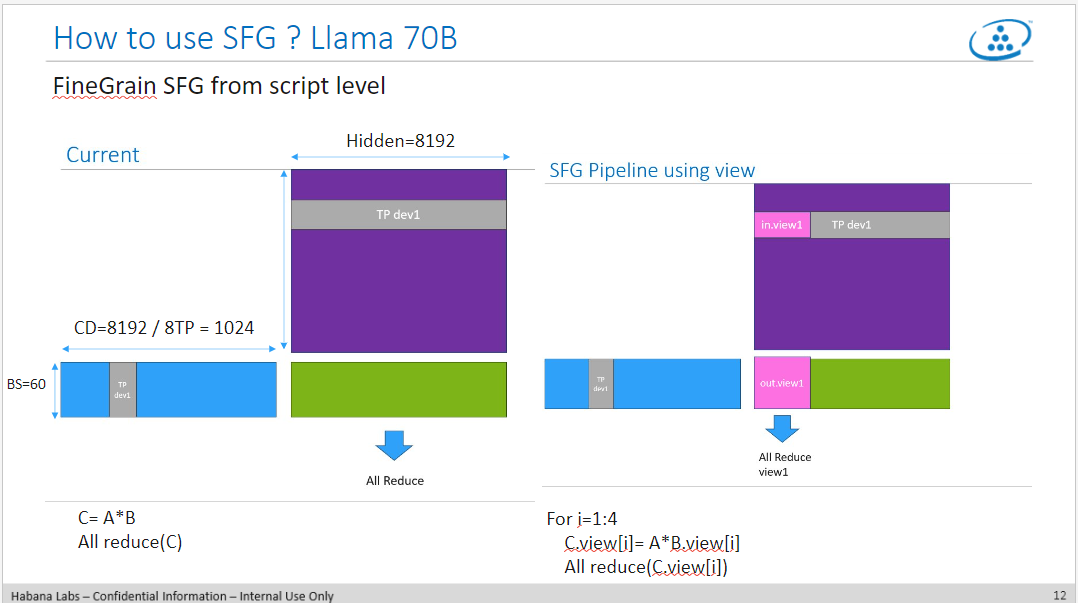
## Design

Framework SFG [Design document](https://habanalabs-my.sharepoint.com/:w:/g/personal/lbachar_habana_ai/EdAgxZDCHA5KgncfTT6ZDd4BkLl3beCfT0iOI18SjG-gkg?email=ssaraswati%40habana.ai&e=8SieA5&isSPOFile=1&sourceId=&params=%7B%22AppName%22%3A%22Teams-Desktop%22%2C%22AppVersion%22%3A%2227%2F22010300409%22%7D)

## Previous Attempts on SFG

The most recent attempt on SFG was to apply SFG on Bloom/Llama inference models on the decode/token generation phase, where the attention layer is already split with TP. The strategy was to split the matmul op into independent slices, and trigger the allreduce as soon as a slice is done, thus creating an opportunity for overlapping the allreduce with the slice computes.

As proposed [here](https://habanalabs.sharepoint.com/:p:/r/sites/ArchitecturePortal/_layouts/15/Doc.aspx?sourcedoc=%7BDA3BC3FA-36CE-411B-98F9-4330248193A4%7D&file=SFG%20for%20GPT3%20and%20Falcon%20v0r3.pptx&action=edit&mobileredirect=true&DefaultItemOpen=1) -



This activity didn't yield desired result, the [slides](https://habanalabs-my.sharepoint.com/:p:/r/personal/eradiano_habana_ai/_layouts/15/Doc.aspx?sourcedoc=%7BED5628D6-65E6-4A3B-8391-6F1868E01003%7D&file=SFG_integration_sync_r0_v0.pptx&wdOrigin=TEAMS-MAGLEV.p2p_ns.rwc&action=edit&mobileredirect=true) describe the issue where for tensor sizes < 4 MB, we can't reach the full utilization of NIC bandwidth. The details are captured in [SW-165974](https://jira.habana-labs.com/browse/SW-165974) - Getting issue details... STATUS

## References:

1. [Liran's design for SFG](https://habanalabs-my.sharepoint.com/:w:/g/personal/lbachar_habana_ai/EdAgxZDCHA5KgncfTT6ZDd4BkLl3beCfT0iOI18SjG-gkg?email=ssaraswati%40habana.ai&e=8SieA5&isSPOFile=1&sourceId=&params=%7B%22AppName%22%3A%22Teams-Desktop%22%2C%22AppVersion%22%3A%2227%2F22010300409%22%7D), [Signal from graph (1).docx](file:///C:\download\attachments\100926802\Signal%20from%20graph%20(1).docx%3fversion=1&modificationDate=1736143622382&api=v2)
2. [Eyal's slides for decode stage SFG on Bloom/Llama inference](https://habanalabs.sharepoint.com/:p:/r/sites/ArchitecturePortal/_layouts/15/Doc.aspx?sourcedoc=%7BDA3BC3FA-36CE-411B-98F9-4330248193A4%7D&file=SFG%20for%20GPT3%20and%20Falcon%20v0r3.pptx&action=edit&mobileredirect=true&DefaultItemOpen=1)
3. [Eyal's recent slides for SFG](https://habanalabs.sharepoint.com/:p:/r/sites/ArchitecturePortal/_layouts/15/Doc.aspx?sourcedoc=%7B10F1B202-3426-4A34-A947-2B676F501DF1%7D&file=SFG_integration_sync_r0_v0.pptx&action=edit&mobileredirect=true&DefaultItemOpen=1)
4. Roman's projection for BS, prompt size and model performance