

Performance Analysis of GPU-accelerated OpenMP Applications using HPCToolkit



John Mellor-Crummey
Rice University

August 25, 2023

OpenMP Users Monthly Telecon



U.S. DEPARTMENT OF
ENERGY

Office of
Science



Outline

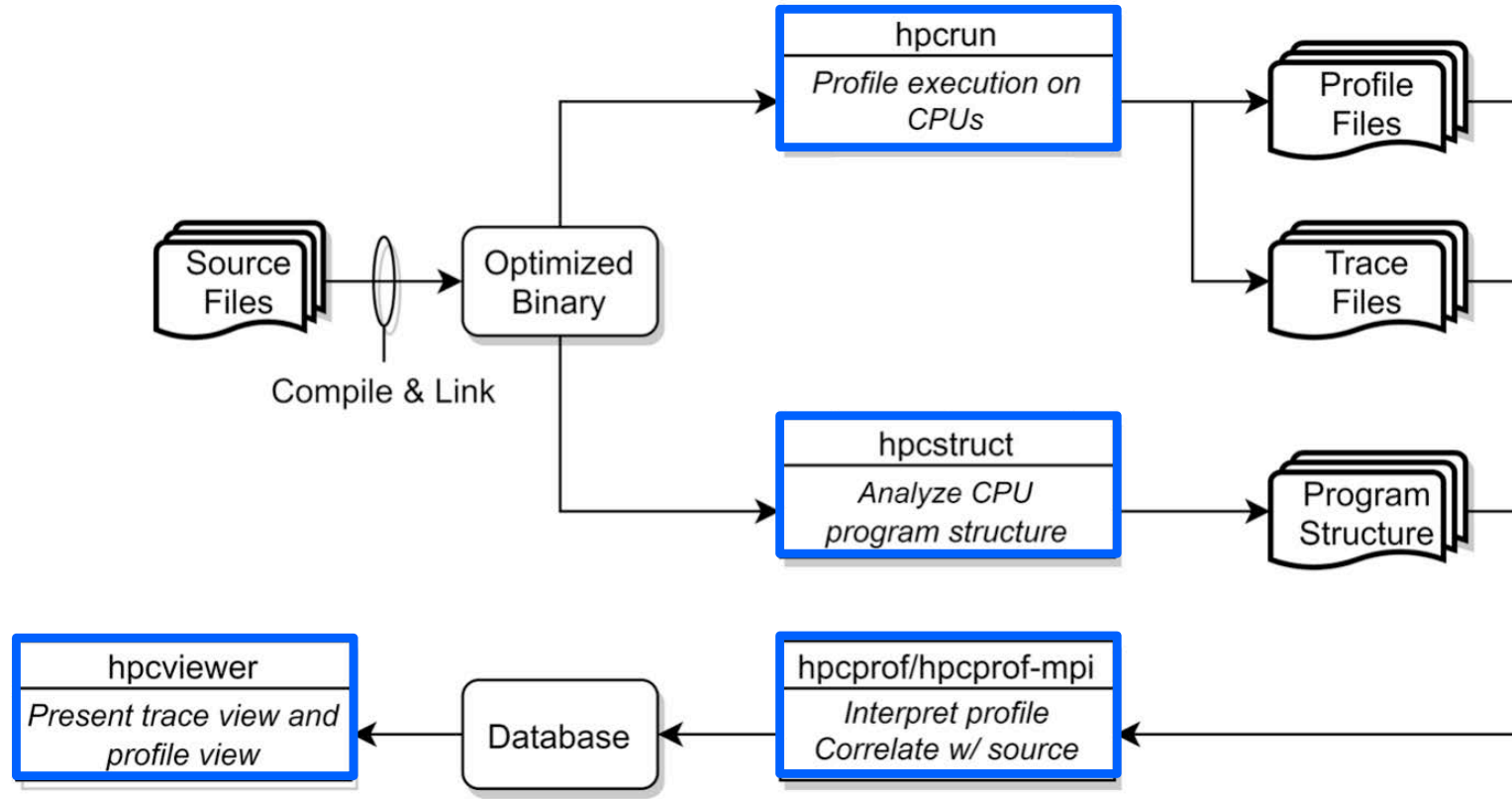
- Introduction to HPCToolkit
 - Overview of HPCToolkit components and their workflow
 - HPCToolkit's graphical user interfaces
- Analyzing the performance of GPU-accelerated codes with HPCToolkit
 - GAMESS
 - GEM
- Status
- Ongoing work

Rice University's HPCToolkit Performance Tools

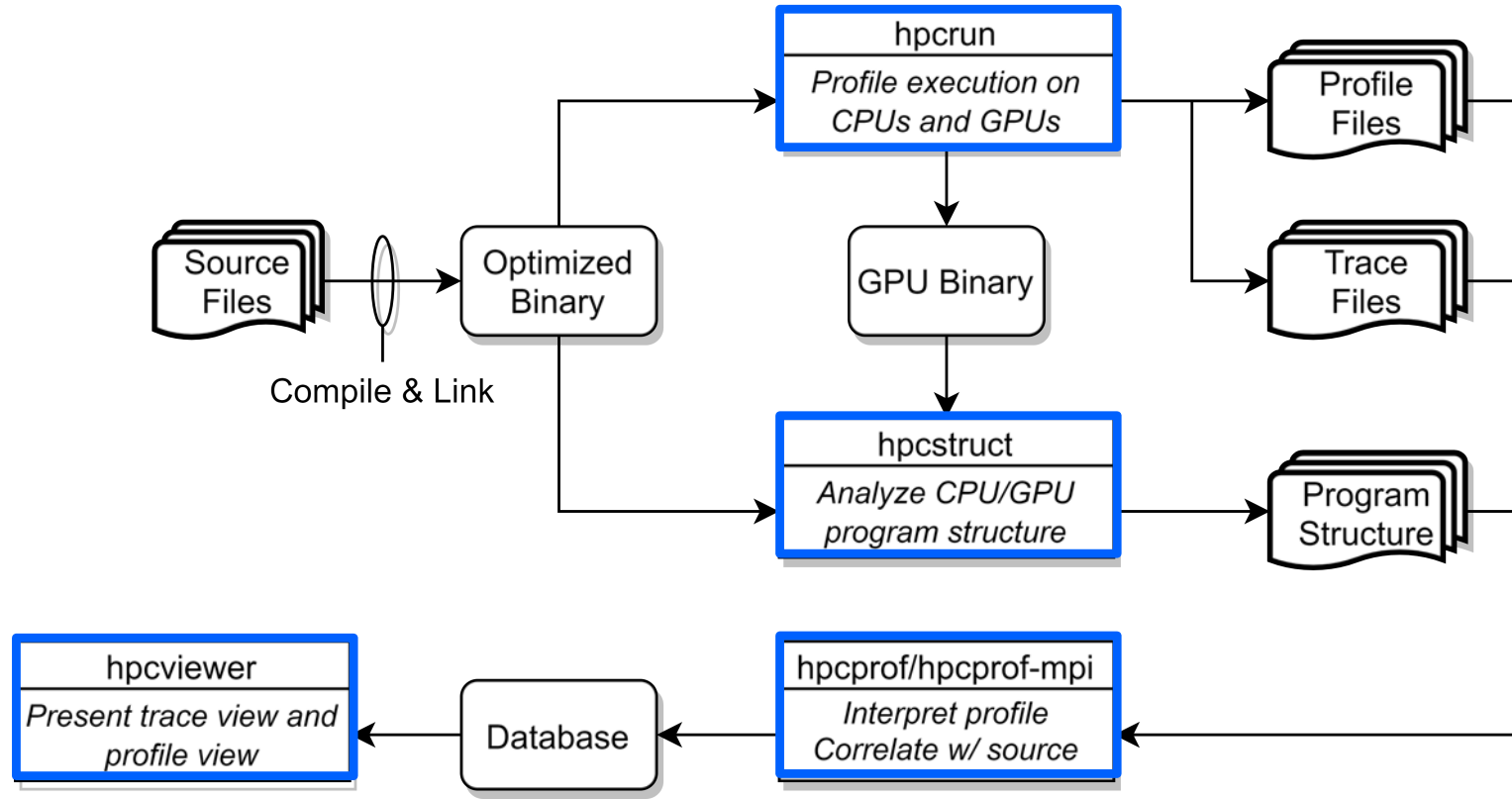
Measure and analyze performance of CPU and GPU-accelerated applications

- Easy: profile unmodified application binaries
- Fast: low-overhead measurement
- Informative: understand where an application spends its time and why
 - call path profiles associate metrics with application source code contexts
 - optional hierarchical traces to understand execution dynamics
- Broad audience
 - application developers
 - framework developers
 - runtime and tool developers

HPCToolkit's Workflow for CPU Applications



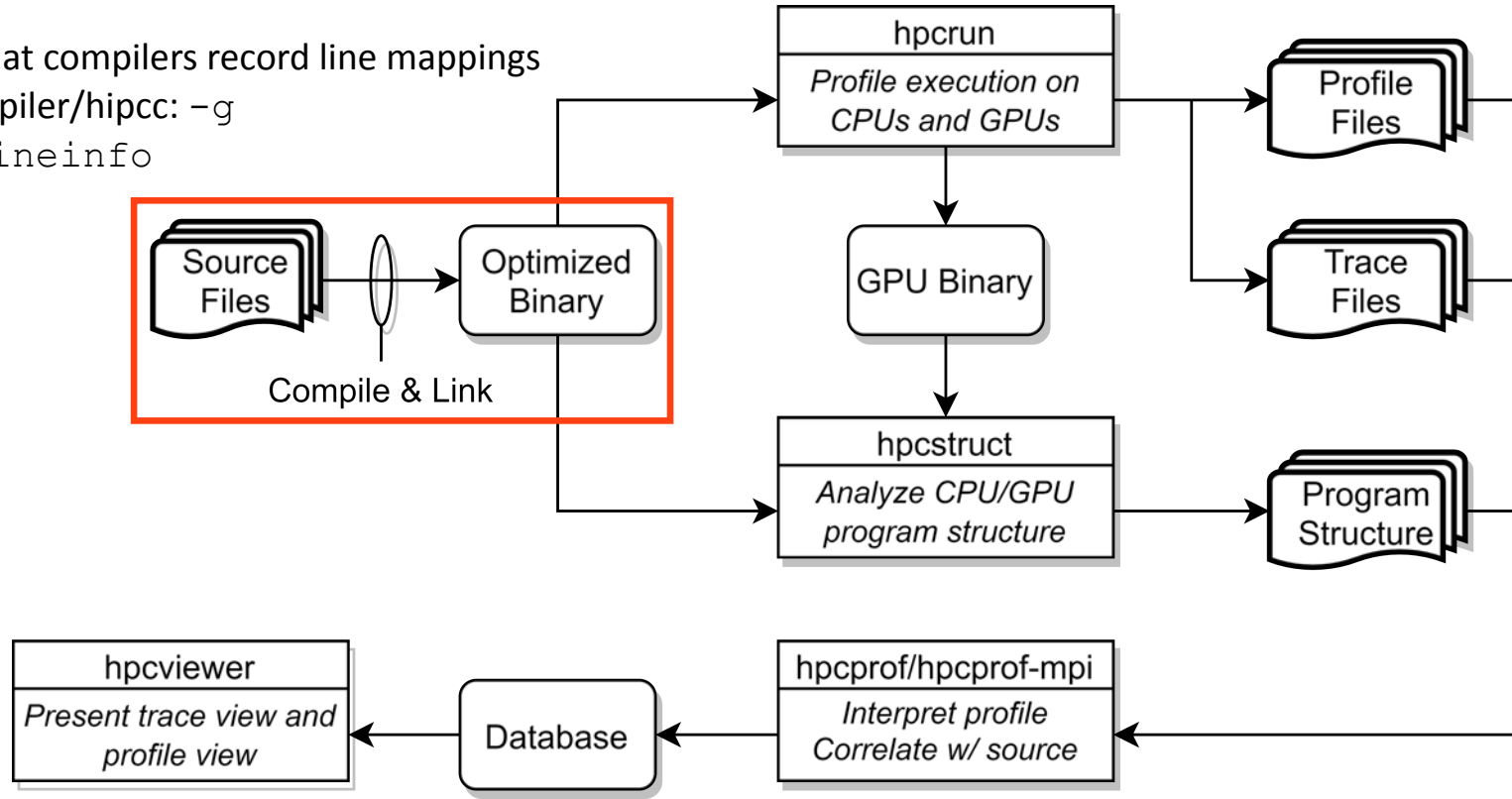
HPCToolkit's Workflow for GPU-accelerated Applications



HPCToolkit's Workflow for GPU-accelerated Applications

Step 1:

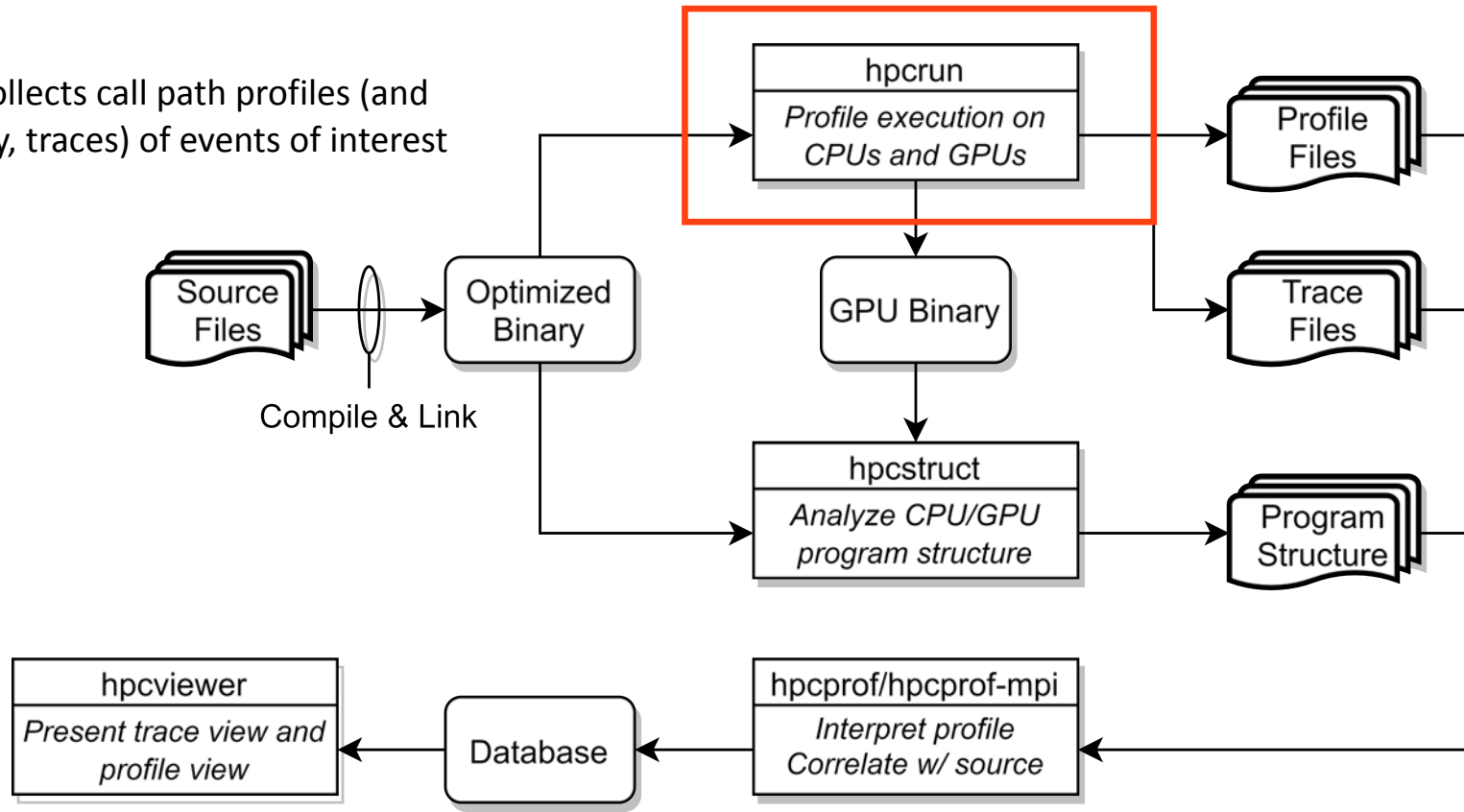
- Ensure that compilers record line mappings
- host compiler/hipcc: -g
- nvcc: -lineinfo



HPCToolkit's Workflow for GPU-accelerated Applications

Step 2:

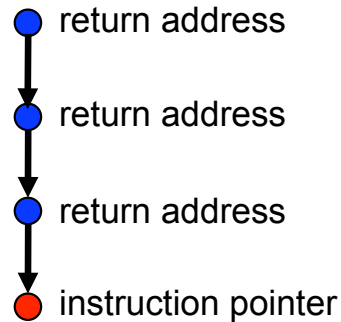
- *hpcrun* collects call path profiles (and optionally, traces) of events of interest



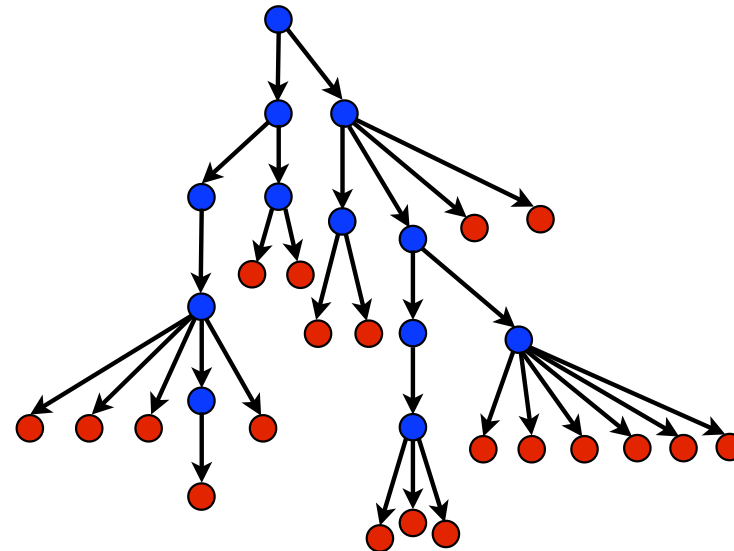
Call Stack Unwinding to Attribute Costs in Context

- Unwind when timer or hardware counter overflows
 - measurement overhead proportional to sampling frequency rather than call frequency
- Unwind to capture context for events such as GPU kernel launches

Call path sample



Calling context tree



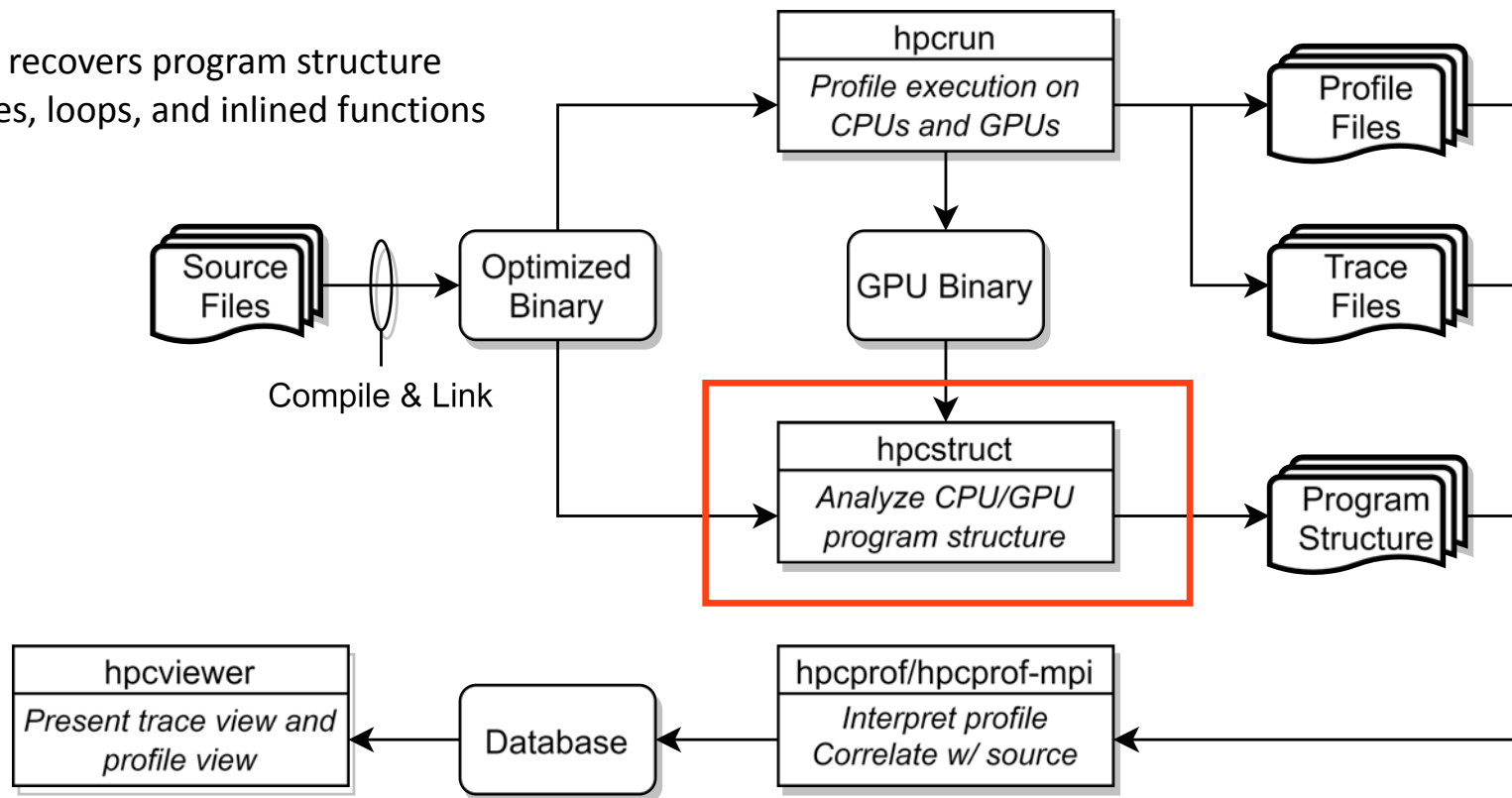
hpcrun: Measure CPU and/or GPU activity

- GPU profiling
 - `hpcrun -e gpu=xxx <app> ...` *xxx* ∈ {nvidia,amd,opencl,level0}
- GPU instrumentation (Intel GPU only)
 - `hpcrun -e gpu=level0,inst=count,latency <app>`
- GPU PC sampling (NVIDIA GPU only)
 - `hpcrun -e gpu=nvidia,pc <app>`
- CPU and GPU Tracing (in addition to profiling)
 - `hpcrun -e CPUTIME -e gpu=xxx -t <app>`
- Use hpcrun with job launchers
 - `jsrun -n 32 -g 1 -a 1 hpcrun -e gpu=xxx <app>`
 - `srun -n 1 -G 1 hpcrun -e gpu=xxx <app>`
 - `aprun -n 16 -N 8 -d 8 hpcrun -e gpu=xxx <app>`

HPCToolkit's Workflow for GPU-accelerated Applications

Step 3:

- *hpcstruct* recovers program structure about lines, loops, and inlined functions



hpcstruct: Analyze CPU and GPU Binaries Using Multiple Threads

- Usage

```
hpcstruct [--gpucfg yes] <measurement-directory>
```

- What it does

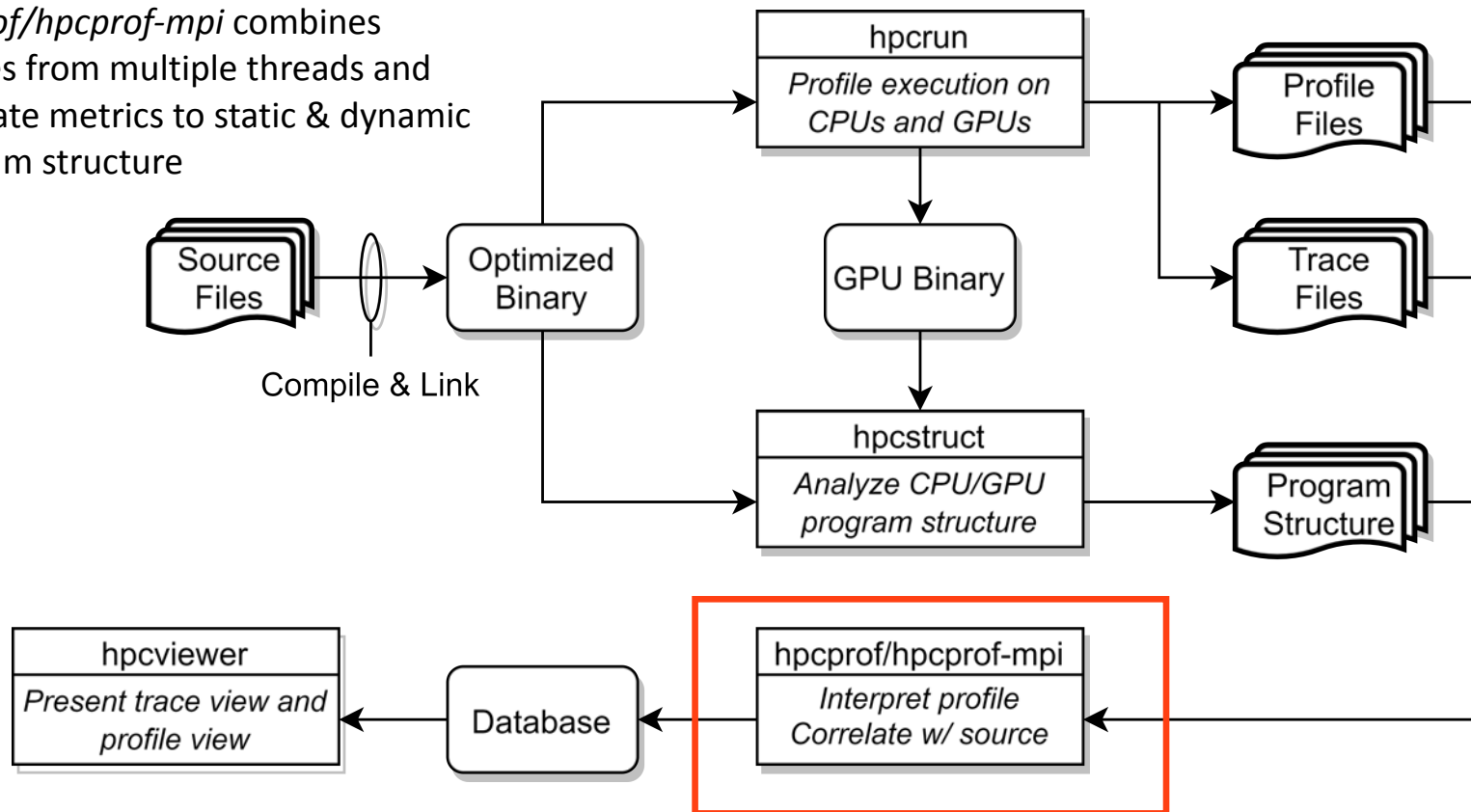
- Recover program structure information
 - Files, functions, inlined templates or functions, loops, source lines
- In parallel, analyze all CPU and GPU binaries that were measured by HPCToolkit
 - default: use size(CPU set)/2 threads
 - analyze large application binaries with 16 threads
 - analyze multiple small application binaries concurrently with 2 threads each
- Cache binary analysis results for reuse when analyzing other executions

NOTE: `--gpucfg yes` needed only for analysis of GPU binaries for interpreting PC samples on NVIDIA GPUs

HPCToolkit's Workflow for GPU-accelerated Applications

Step 4:

- *hpcprof/hpcprof-mpi* combines profiles from multiple threads and correlate metrics to static & dynamic program structure



hpcprof/hpcprof-mpi: Associate Measurements with Program Structure

- Analyze data from modest executions with multithreading

```
hpcprof <measurement-directory>
```

- Analyze data from large executions with distributed-memory parallelism + multithreading

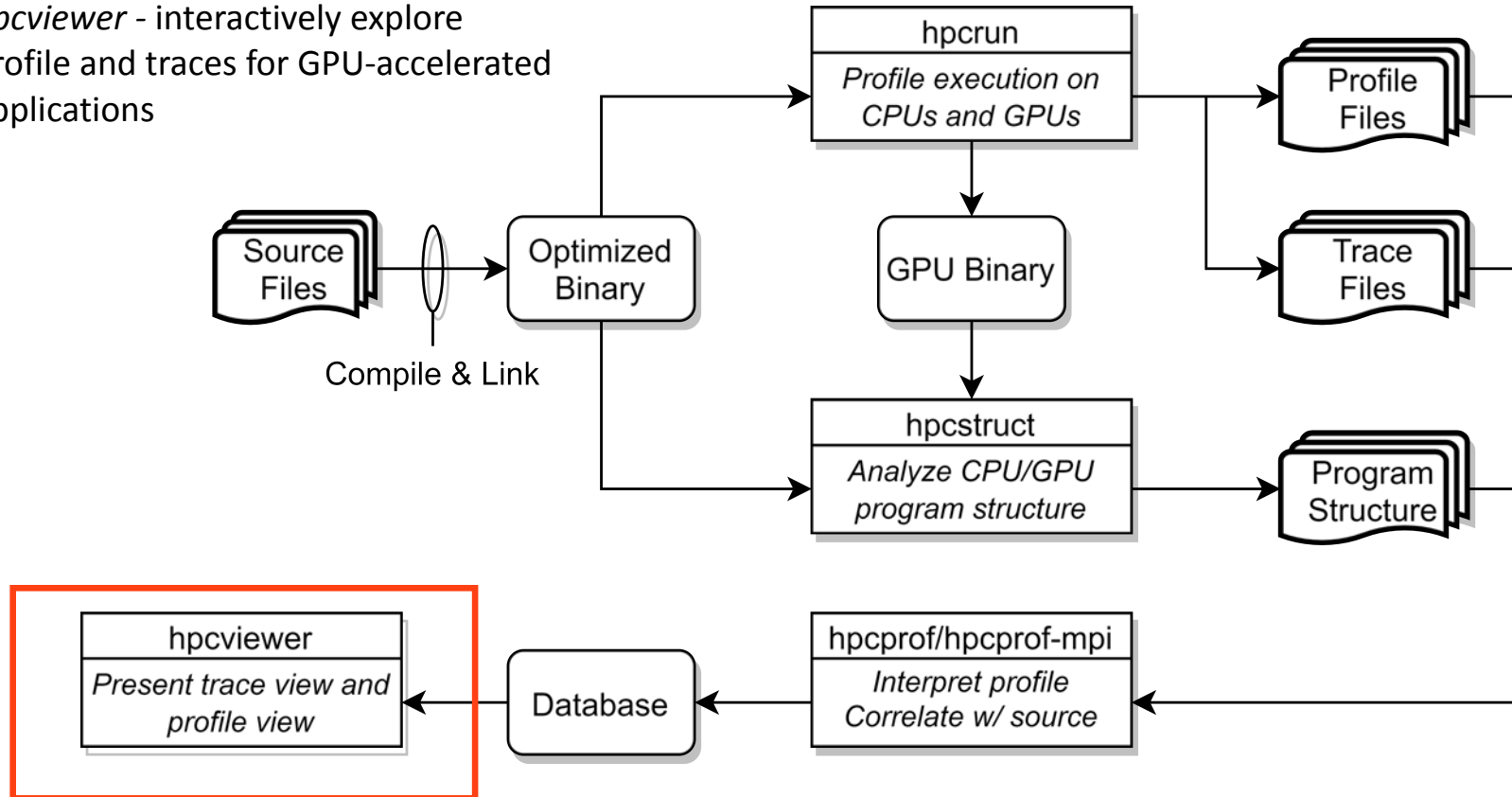
```
jsrun -n 2 -a 1 -c 22 -b packed hpcprof-mpi <measurement-directory>
```

```
srun -N 2 -n 2 -c 126 hpcprof-mpi <measurement-directory>
```

HPCToolkit's Workflow for GPU-accelerated Applications

Step 4:

- *hpcviewer* - interactively explore profile and traces for GPU-accelerated applications



Code-centric Analysis with hpcviewer

The screenshot displays the hpcviewer application interface. The top section shows the source code for 'lulesh-RAJA-parallel.cxx'. The 'view control' section includes buttons for 'Top-down view', 'Bottom-up view', and 'Flat view'. The 'metric display' section shows a tree view of the code structure. The 'navigation pane' is a vertical bar on the right side. The 'metric pane' is a table on the right side showing performance metrics.

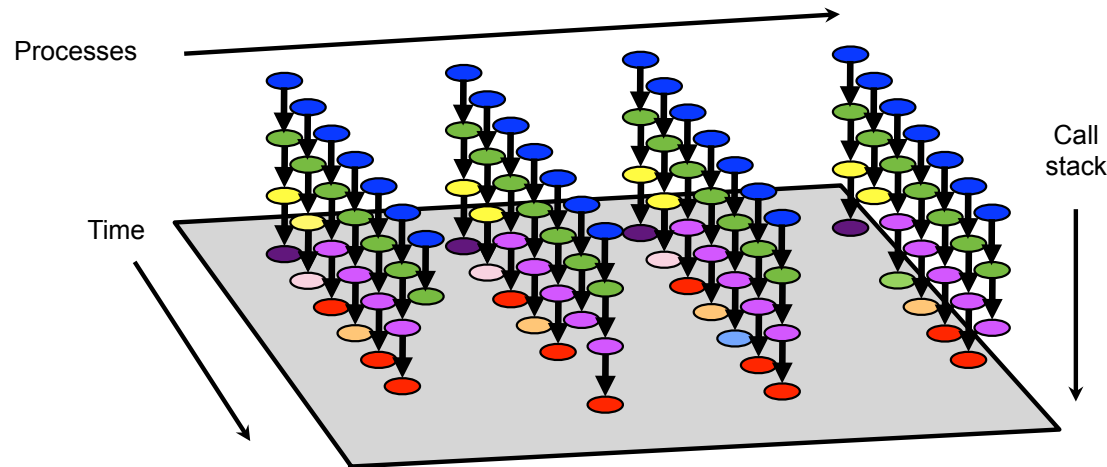
source pane

```

1183
1184 /*****
1185 /* compute the hourglass modes */
1186
1187 RAJA::forall<elem_exec_policy>(*domElemList, [&] (int i2) {
1188 #ifndef OMP_HACK
1189     Real_t hgfx[8], hgyf[8], hgfb[8];
1190 #endif
1191     Real_t coefficient;
1192
1193     Real_t hourgam0[4], hourgam1[4], hourgam2[4], hourgam3[4];
1194     Real_t hourgam4[4], hourgam5[4], hourgam6[4], hourgam7[4];
1195     Real_t xd1[8], yd1[8], zd1[8];
1196
1197     Index_p elemToNode = &nodeList[8*i2];
1198     Index_t i3=8*i2;
1199     Real_t volInv=Real_t(1.0)/determinant[i2];
1200
1201     // compute the hourglass modes
1202     for (int i=0; i<8; i++) {
1203         Real_t x1=x[i], y1=y[i], z1=z[i];
1204         Real_t x2=x[i+8], y2=y[i+8], z2=z[i+8];
1205         Real_t x3=x[i+16], y3=y[i+16], z3=z[i+16];
1206         Real_t x4=x[i+24], y4=y[i+24], z4=z[i+24];
1207         Real_t x5=x[i+32], y5=y[i+32], z5=z[i+32];
1208         Real_t x6=x[i+40], y6=y[i+40], z6=z[i+40];
1209         Real_t x7=x[i+48], y7=y[i+48], z7=z[i+48];
1210         Real_t x8=x[i+56], y8=y[i+56], z8=z[i+56];
1211         Real_t x9=x[i+64], y9=y[i+64], z9=z[i+64];
1212         Real_t x10=x[i+72], y10=y[i+72], z10=z[i+72];
1213         Real_t x11=x[i+80], y11=y[i+80], z11=z[i+80];
1214         Real_t x12=x[i+88], y12=y[i+88], z12=z[i+88];
1215         Real_t x13=x[i+96], y13=y[i+96], z13=z[i+96];
1216         Real_t x14=x[i+104], y14=y[i+104], z14=z[i+104];
1217         Real_t x15=x[i+112], y15=y[i+112], z15=z[i+112];
1218         Real_t x16=x[i+120], y16=y[i+120], z16=z[i+120];
1219         Real_t x17=x[i+128], y17=y[i+128], z17=z[i+128];
1220         Real_t x18=x[i+136], y18=y[i+136], z18=z[i+136];
1221         Real_t x19=x[i+144], y19=y[i+144], z19=z[i+144];
1222         Real_t x20=x[i+152], y20=y[i+152], z20=z[i+152];
1223         Real_t x21=x[i+160], y21=y[i+160], z21=z[i+160];
1224         Real_t x22=x[i+168], y22=y[i+168], z22=z[i+168];
1225         Real_t x23=x[i+176], y23=y[i+176], z23=z[i+176];
1226         Real_t x24=x[i+184], y24=y[i+184], z24=z[i+184];
1227         Real_t x25=x[i+192], y25=y[i+192], z25=z[i+192];
1228         Real_t x26=x[i+200], y26=y[i+200], z26=z[i+200];
1229         Real_t x27=x[i+208], y27=y[i+208], z27=z[i+208];
1230         Real_t x28=x[i+216], y28=y[i+216], z28=z[i+216];
1231         Real_t x29=x[i+224], y29=y[i+224], z29=z[i+224];
1232         Real_t x30=x[i+232], y30=y[i+232], z30=z[i+232];
1233         Real_t x31=x[i+240], y31=y[i+240], z31=z[i+240];
1234         Real_t x32=x[i+248], y32=y[i+248], z32=z[i+248];
1235         Real_t x33=x[i+256], y33=y[i+256], z33=z[i+256];
1236         Real_t x34=x[i+264], y34=y[i+264], z34=z[i+264];
1237         Real_t x35=x[i+272], y35=y[i+272], z35=z[i+272];
1238         Real_t x36=x[i+280], y36=y[i+280], z36=z[i+280];
1239         Real_t x37=x[i+288], y37=y[i+288], z37=z[i+288];
1240         Real_t x38=x[i+296], y38=y[i+296], z38=z[i+296];
1241         Real_t x39=x[i+304], y39=y[i+304], z39=z[i+304];
1242         Real_t x40=x[i+312], y40=y[i+312], z40=z[i+312];
1243         Real_t x41=x[i+320], y41=y[i+320], z41=z[i+320];
1244         Real_t x42=x[i+328], y42=y[i+328], z42=z[i+328];
1245         Real_t x43=x[i+336], y43=y[i+336], z43=z[i+336];
1246         Real_t x44=x[i+344], y44=y[i+344], z44=z[i+344];
1247         Real_t x45=x[i+352], y45=y[i+352], z45=z[i+352];
1248         Real_t x46=x[i+360], y46=y[i+360], z46=z[i+360];
1249         Real_t x47=x[i+368], y47=y[i+368], z47=z[i+368];
1250         Real_t x48=x[i+376], y48=y[i+376], z48=z[i+376];
1251         Real_t x49=x[i+384], y49=y[i+384], z49=z[i+384];
1252         Real_t x50=x[i+392], y50=y[i+392], z50=z[i+392];
1253         Real_t x51=x[i+400], y51=y[i+400], z51=z[i+400];
1254         Real_t x52=x[i+408], y52=y[i+408], z52=z[i+408];
1255         Real_t x53=x[i+416], y53=y[i+416], z53=z[i+416];
1256         Real_t x54=x[i+424], y54=y[i+424], z54=z[i+424];
1257         Real_t x55=x[i+432], y55=y[i+432], z55=z[i+432];
1258         Real_t x56=x[i+440], y56=y[i+440], z56=z[i+440];
1259         Real_t x57=x[i+448], y57=y[i+448], z57=z[i+448];
1260         Real_t x58=x[i+456], y58=y[i+456], z58=z[i+456];
1261         Real_t x59=x[i+464], y59=y[i+464], z59=z[i+464];
1262         Real_t x60=x[i+472], y60=y[i+472], z60=z[i+472];
1263         Real_t x61=x[i+480], y61=y[i+480], z61=z[i+480];
1264         Real_t x62=x[i+488], y62=y[i+488], z62=z[i+488];
1265         Real_t x63=x[i+496], y63=y[i+496], z63=z[i+496];
1266         Real_t x64=x[i+504], y64=y[i+504], z64=z[i+504];
1267         Real_t x65=x[i+512], y65=y[i+512], z65=z[i+512];
1268         Real_t x66=x[i+520], y66=y[i+520], z66=z[i+520];
1269         Real_t x67=x[i+528], y67=y[i+528], z67=z[i+528];
1270         Real_t x68=x[i+536], y68=y[i+536], z68=z[i+536];
1271         Real_t x69=x[i+544], y69=y[i+544], z69=z[i+544];
1272         Real_t x70=x[i+552], y70=y[i+552], z70=z[i+552];
1273         Real_t x71=x[i+560], y71=y[i+560], z71=z[i+560];
1274         Real_t x72=x[i+568], y72=y[i+568], z72=z[i+568];
1275         Real_t x73=x[i+576], y73=y[i+576], z73=z[i+576];
1276         Real_t x74=x[i+584], y74=y[i+584], z74=z[i+584];
1277         Real_t x75=x[i+592], y75=y[i+592], z75=z[i+592];
1278         Real_t x76=x[i+600], y76=y[i+600], z76=z[i+600];
1279         Real_t x77=x[i+608], y77=y[i+608], z77=z[i+608];
1280         Real_t x78=x[i+616], y78=y[i+616], z78=z[i+616];
1281         Real_t x79=x[i+624], y79=y[i+624], z79=z[i+624];
1282         Real_t x80=x[i+632], y80=y[i+632], z80=z[i+632];
1283         Real_t x81=x[i+640], y81=y[i+640], z81=z[i+640];
1284         Real_t x82=x[i+648], y82=y[i+648], z82=z[i+648];
1285         Real_t x83=x[i+656], y83=y[i+656], z83=z[i+656];
1286         Real_t x84=x[i+664], y84=y[i+664], z84=z[i+664];
1287         Real_t x85=x[i+672], y85=y[i+672], z85=z[i+672];
1288         Real_t x86=x[i+680], y86=y[i+680], z86=z[i+680];
1289         Real_t x87=x[i+688], y87=y[i+688], z87=z[i+688];
1290         Real_t x88=x[i+696], y88=y[i+696], z88=z[i+696];
1291         Real_t x89=x[i+704], y89=y[i+704], z89=z[i+704];
1292         Real_t x90=x[i+712], y90=y[i+712], z90=z[i+712];
1293         Real_t x91=x[i+720], y91=y[i+720], z91=z[i+720];
1294         Real_t x92=x[i+728], y92=y[i+728], z92=z[i+728];
1295         Real_t x93=x[i+736], y93=y[i+736], z93=z[i+736];
1296         Real_t x94=x[i+744], y94=y[i+744], z94=z[i+744];
1297         Real_t x95=x[i+752], y95=y[i+752], z95=z[i+752];
1298         Real_t x96=x[i+760], y96=y[i+760], z96=z[i+760];
1299         Real_t x97=x[i+768], y97=y[i+768], z97=z[i+768];
1300         Real_t x98=x[i+776], y98=y[i+776], z98=z
```

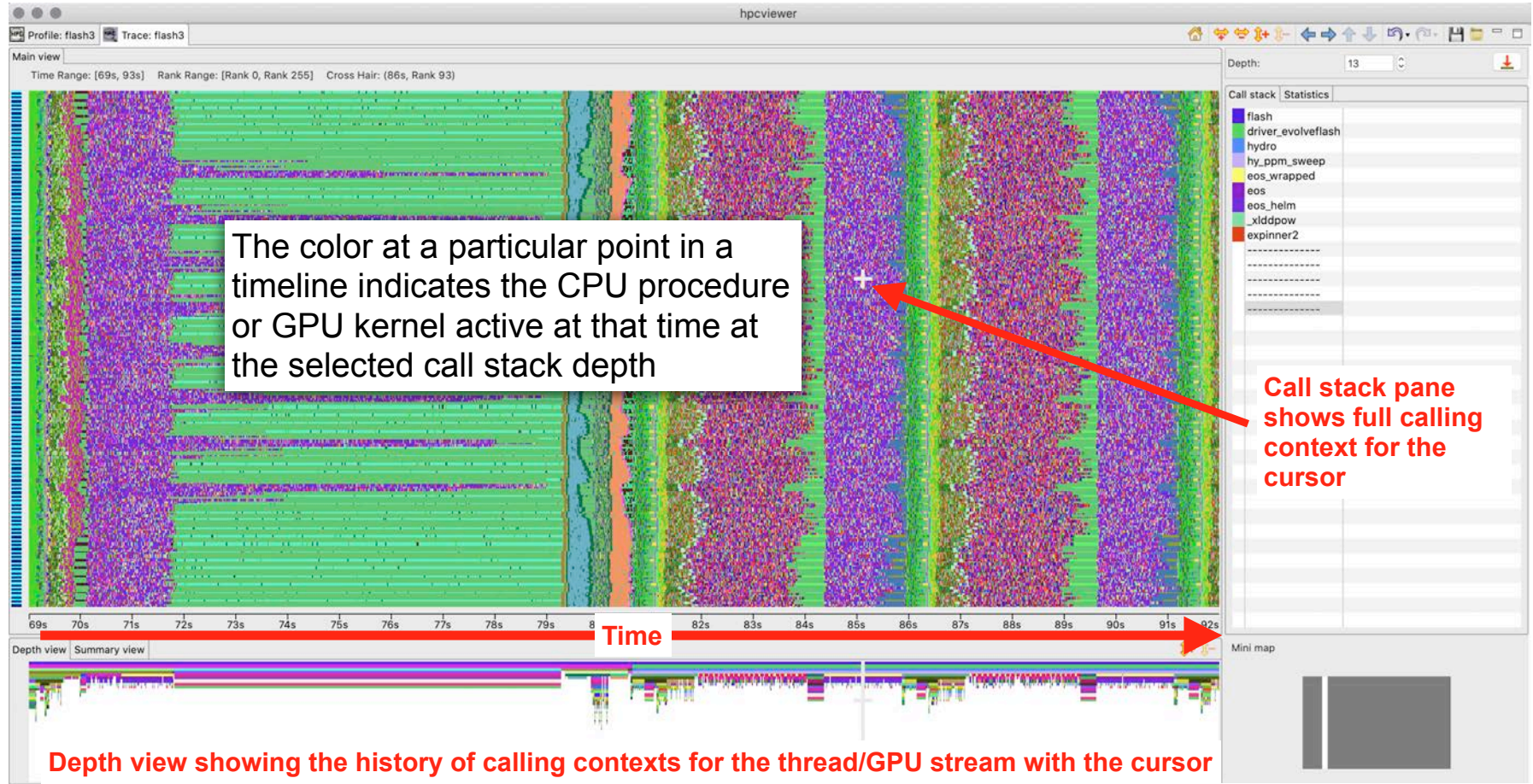
Understanding Temporal Behavior

- Profiling compresses out the temporal dimension
 - Temporal patterns, e.g. serial sections and dynamic load imbalance are invisible in profiles
- What can we do? Trace call path samples
 - N times per second, take a call path sample of each thread
 - Organize the samples for each thread along a time line
 - View how the execution evolves left to right
 - What do we view? assign each procedure a color; view a depth slice of an execution

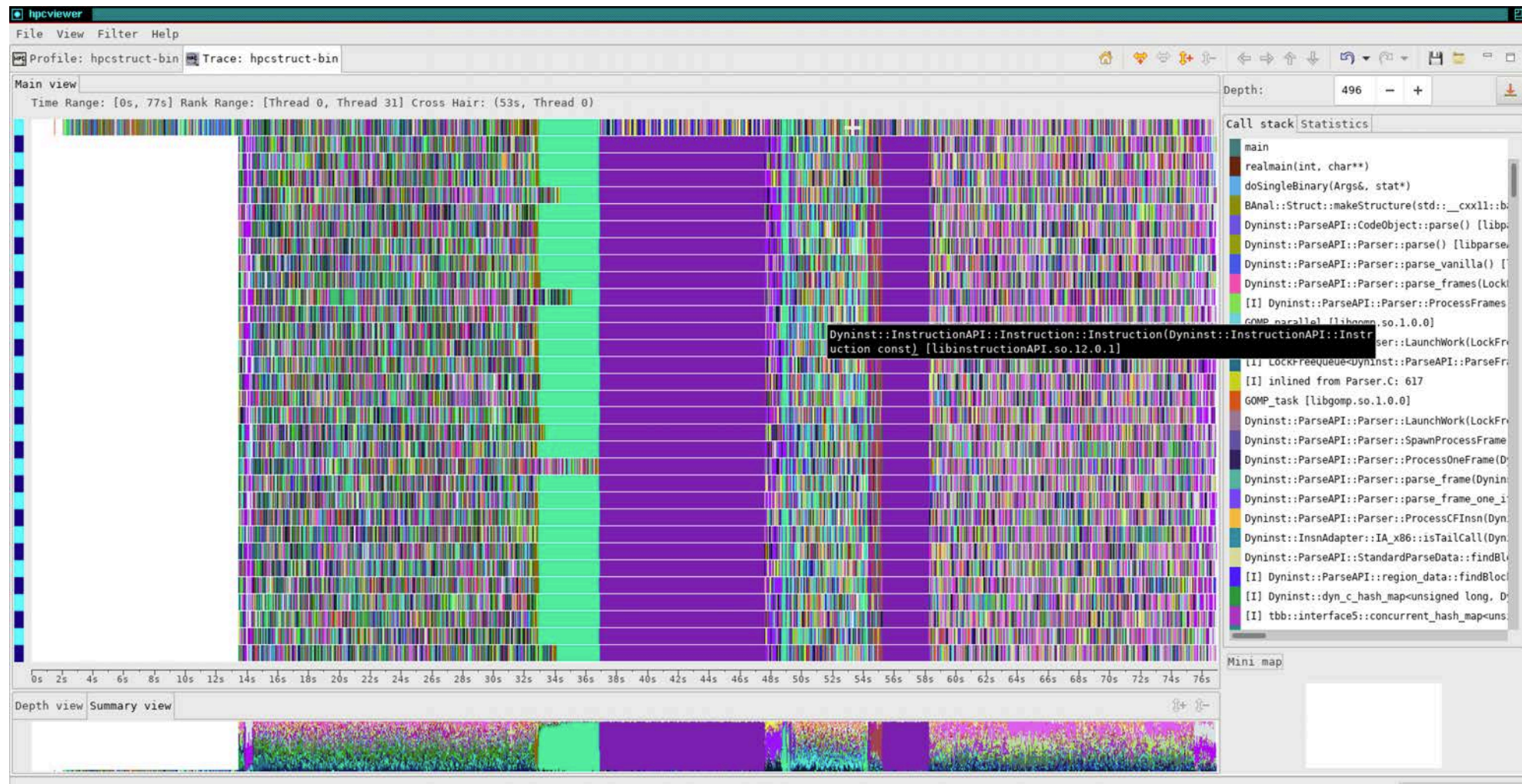


Time-centric Analysis with hpcviewer

MPI ranks, OpenMP Threads, GPU streams



hpcstruct Example: Analyze 7.7GB TensorFlow library (170MB text) in 77s



Case Studies

- GAMESS - an ab initio quantum chemistry package: Fortran + MPI + OpenMP offloading
- GEM - a gyrokinetic turbulence code that simulates both ions and electrons

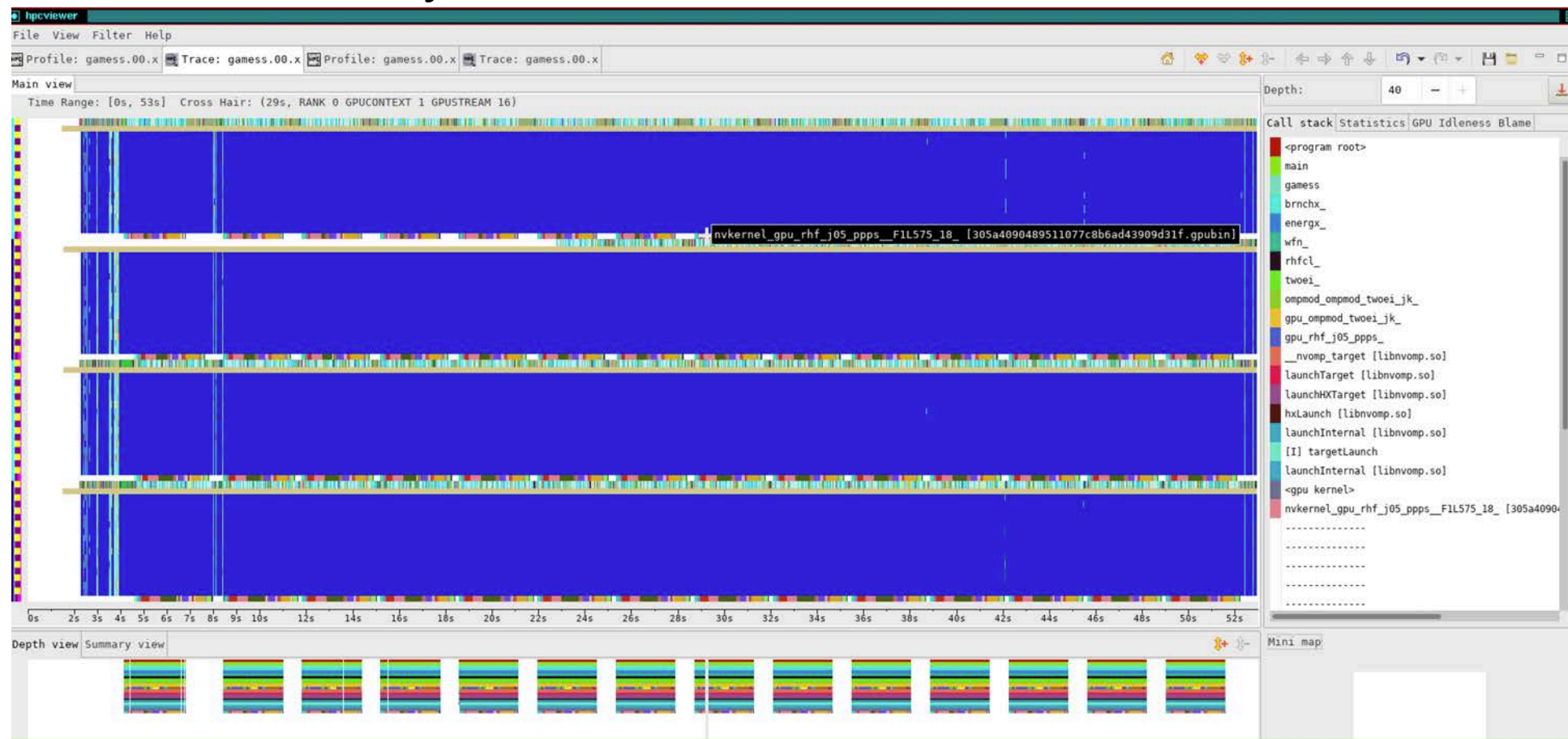
Case Study: GAMESS

- General Atomic and Molecular Electronic Structure System (GAMESS)
 - general *ab initio* quantum chemistry package
- Calculates the energies, structures, and properties of a wide range of chemical systems
- Experiments
 - GPU-accelerated nodes at a Perlmutter hackathon
 - Single node with 4 GPUs
 - Five nodes with 20 GPUs

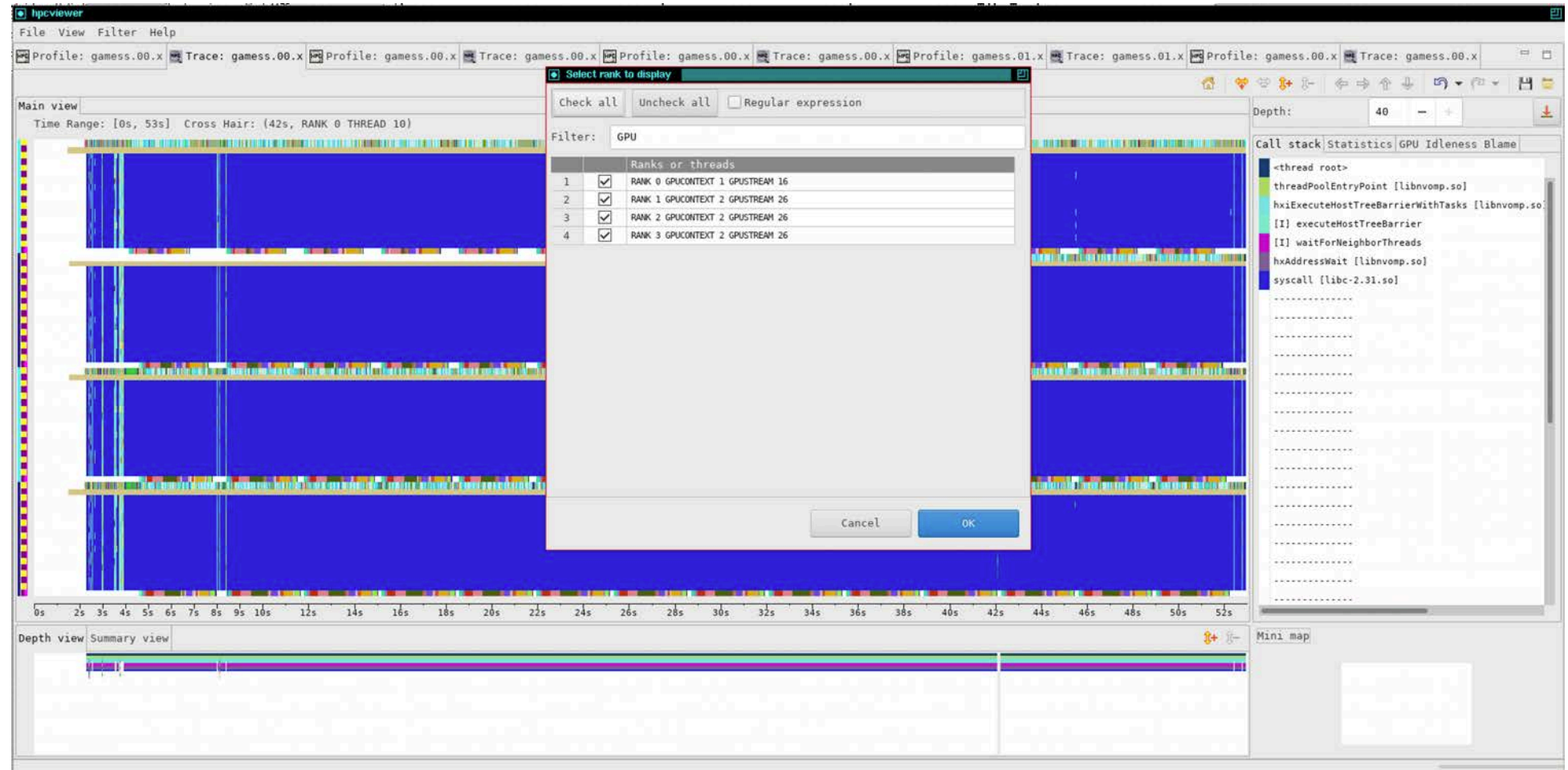
Perlmutter node at a glance

AMD Milan CPU
4 NVIDIA A100 GPUs
256 GB memory

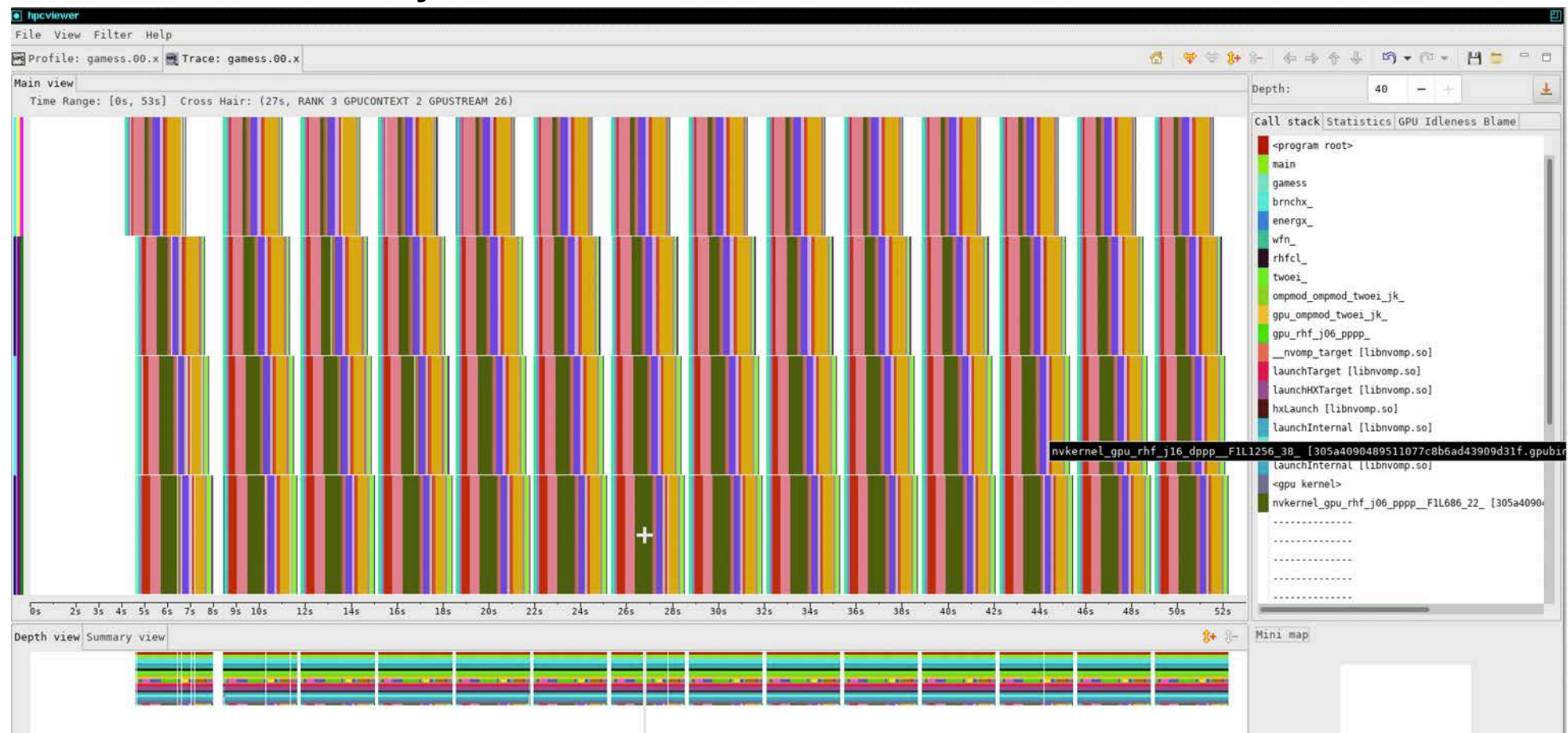
Time-centric Analysis: GAMESS 4 ranks, 4 GPUs on Perlmutter



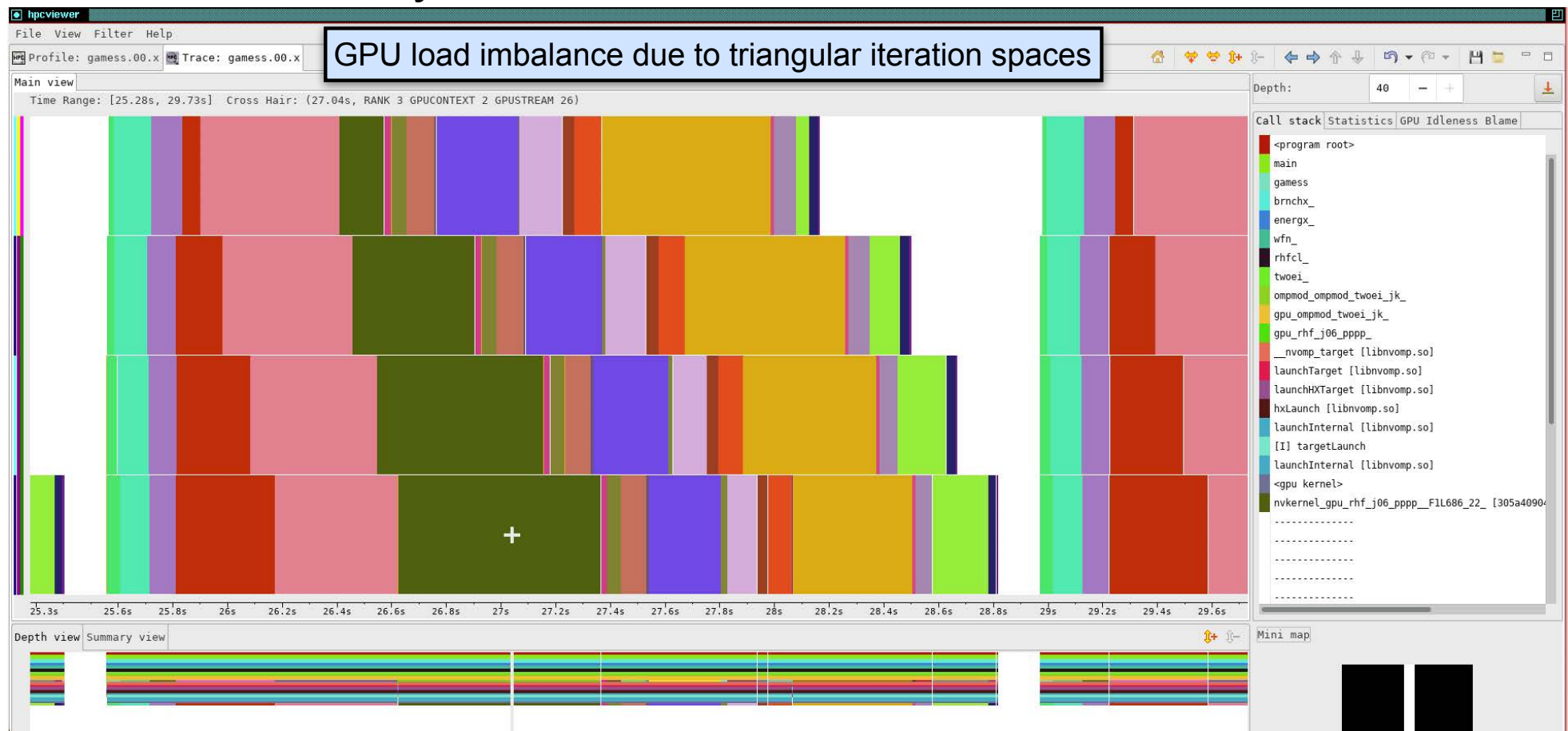
Time-centric Analysis: GAMESS 4 ranks, 4 GPUs on Perlmutter



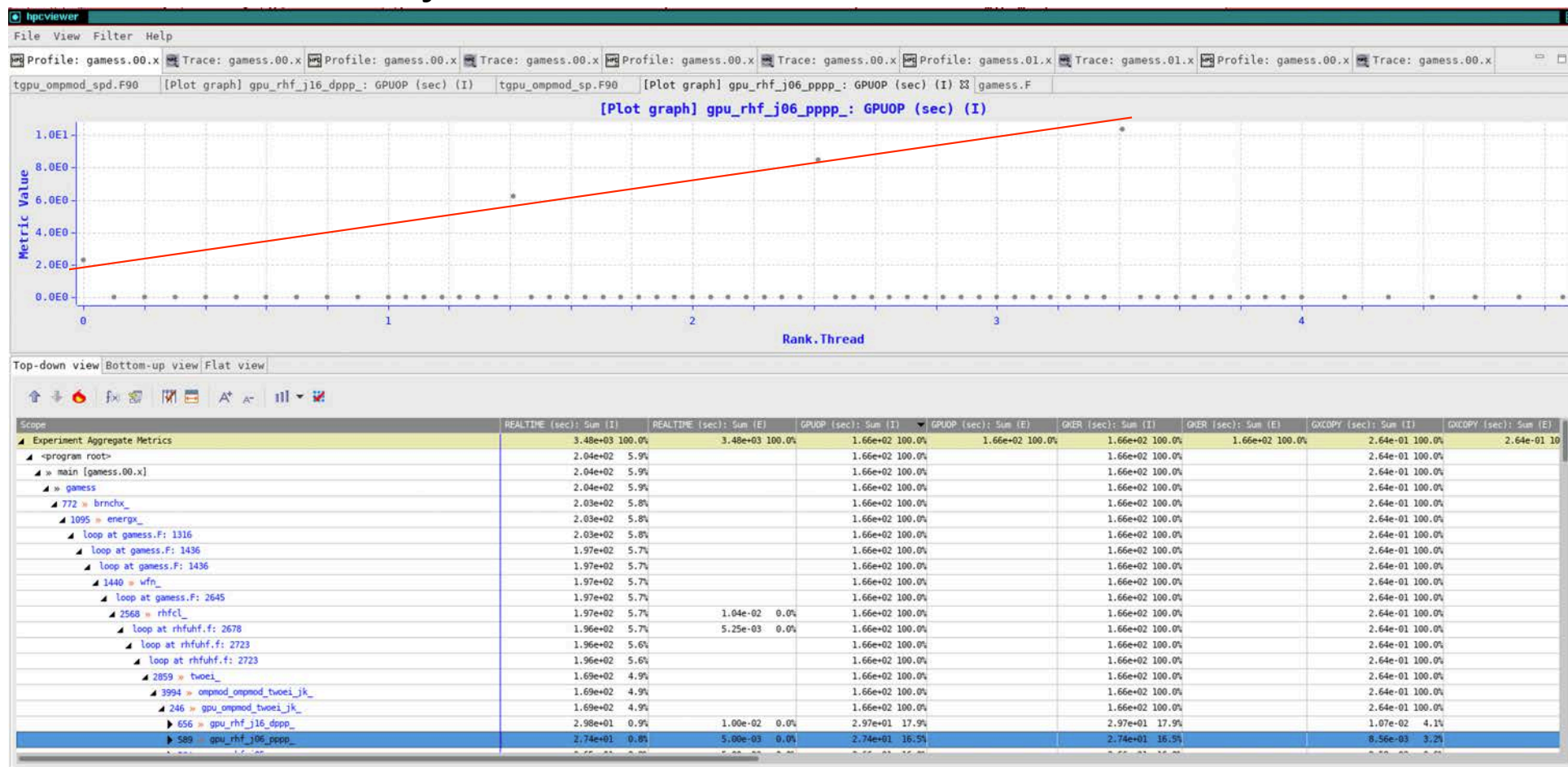
Time-centric Analysis: GAMESS 4 ranks, 4 GPUs on Perlmutter



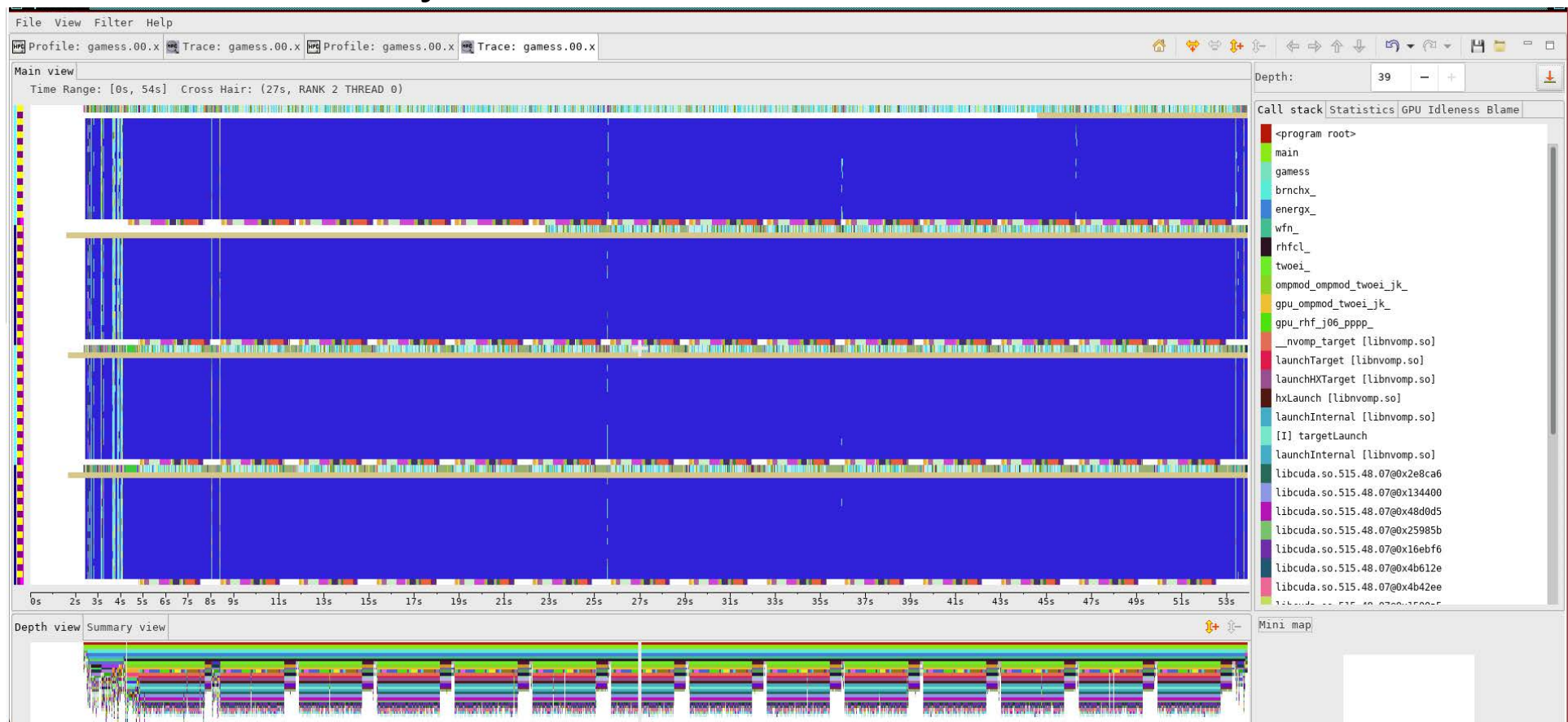
Time-centric Analysis: GAMESS 4 ranks, 4 GPUs on Perlmutter



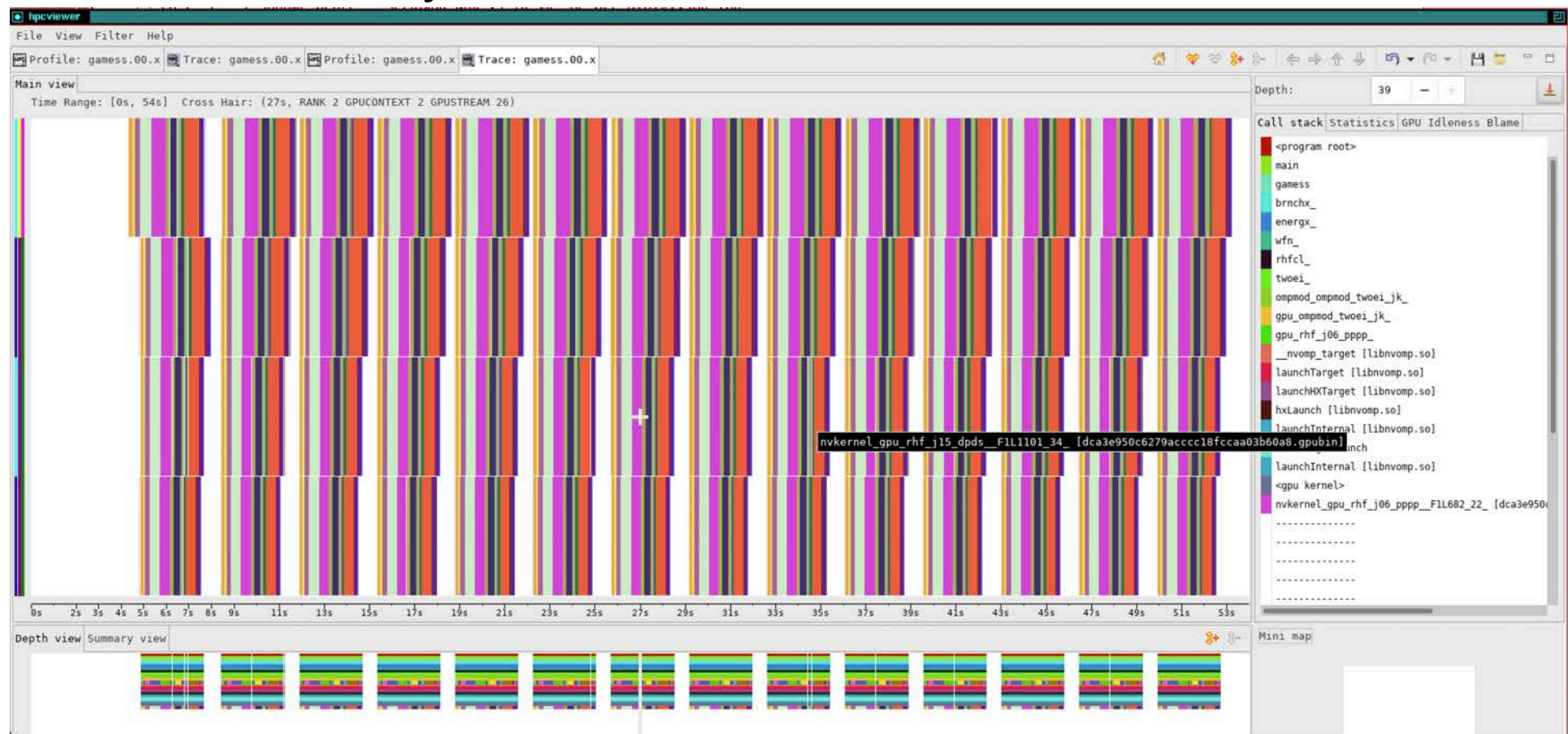
Time-centric Analysis: GAMESS 4 ranks, 4 GPUs on Perlmutter



Time-centric Analysis: GAMESS 4 ranks, 4 GPUs on Perlmutter



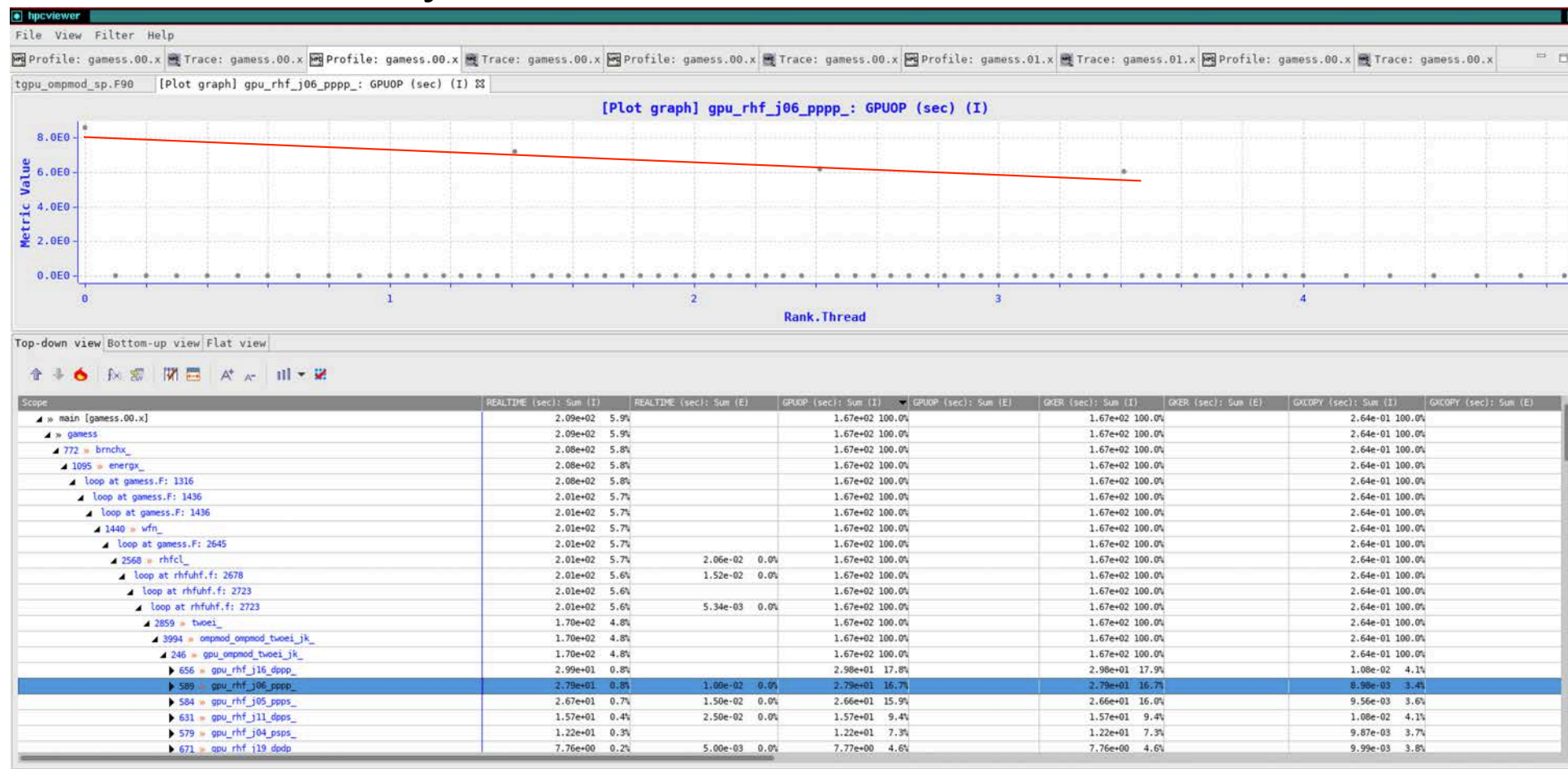
Time-centric Analysis: GAMESS 4 ranks, 4 GPUs on Perlmutter



Time-centric Analysis: GAMESS 4 ranks, 4 GPUs on Perlmutter



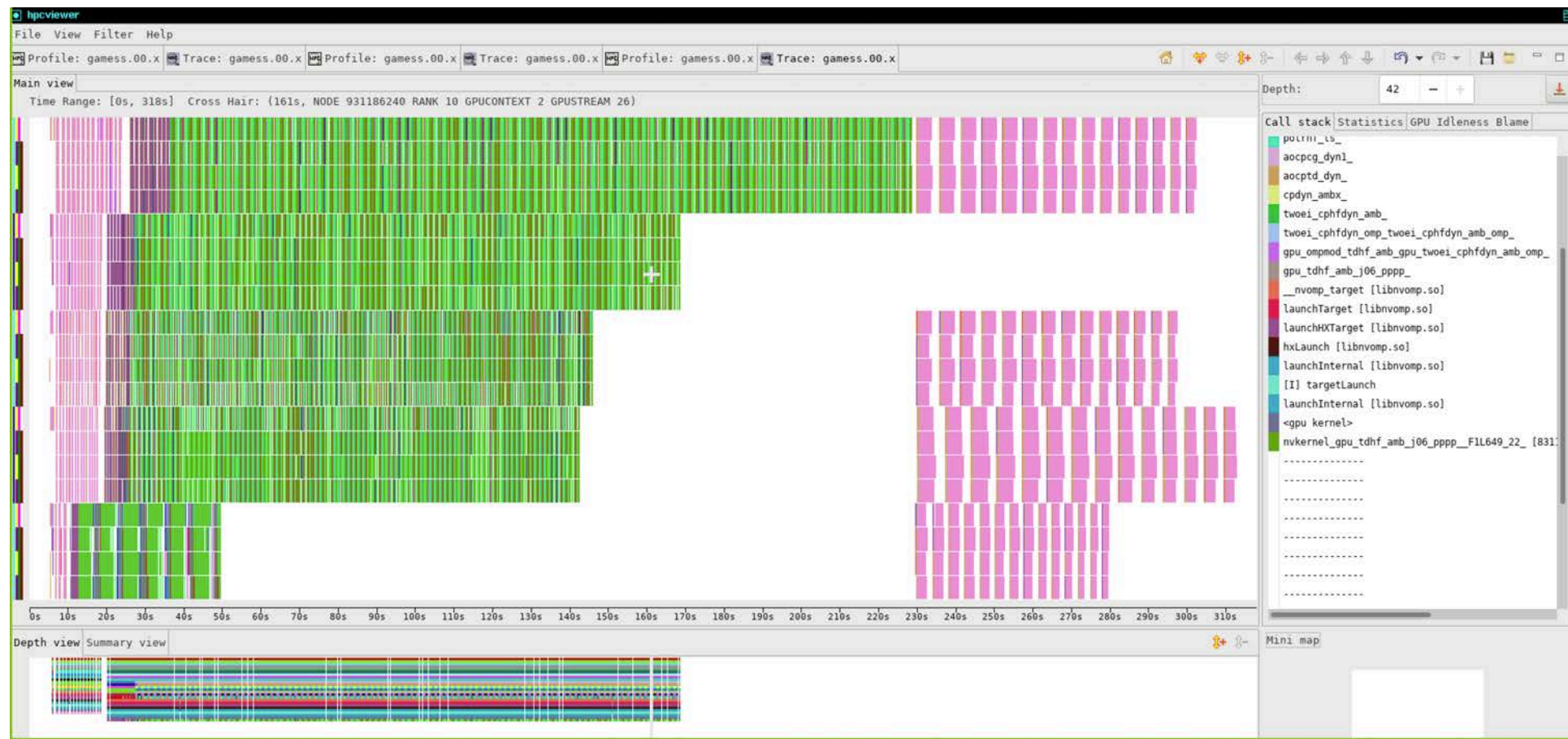
Time-centric Analysis: GAMESS 4 ranks, 4 GPUs on Perlmutter



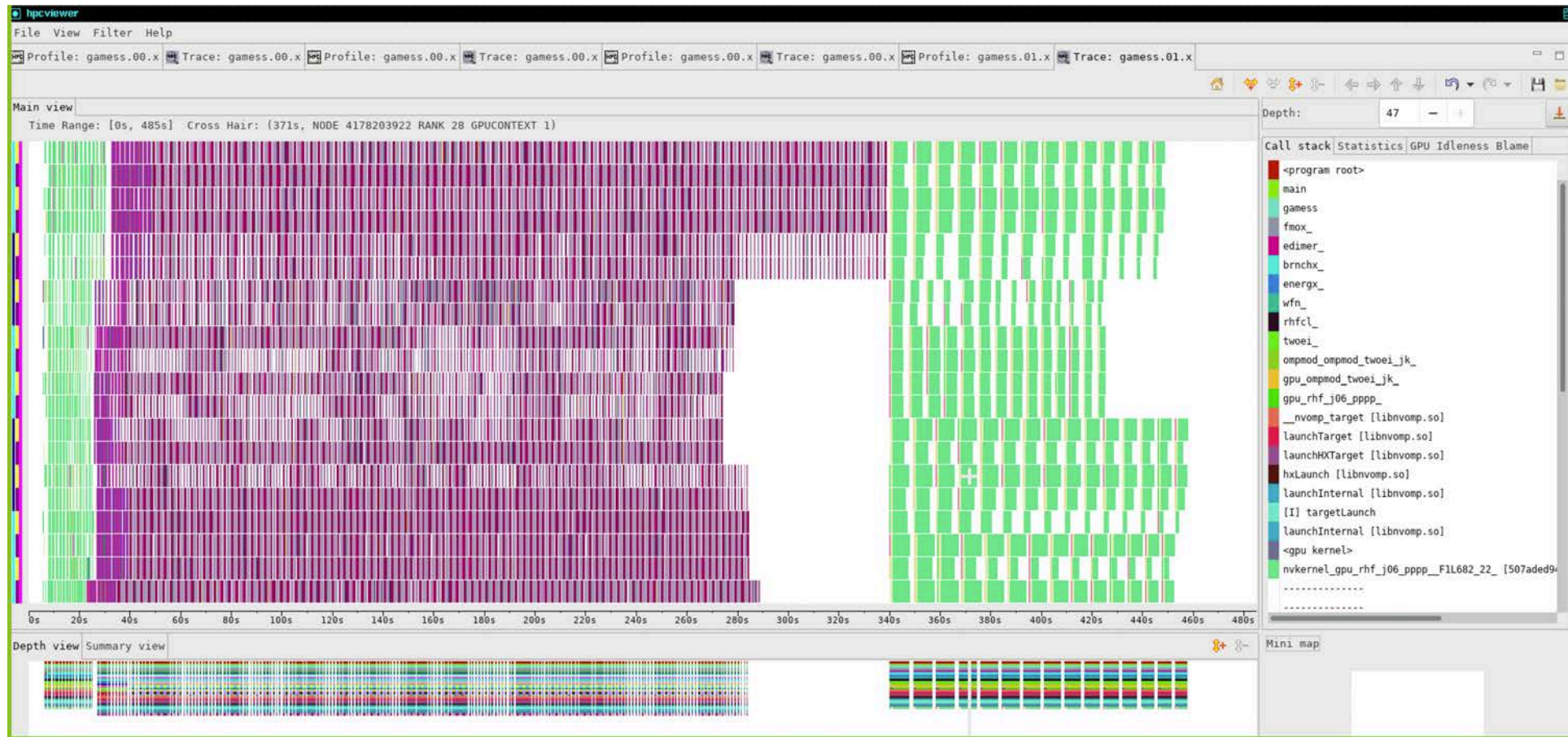
Time-centric Analysis: GAMESS 5 nodes, 40 ranks, 20 GPUs on Perlmutter



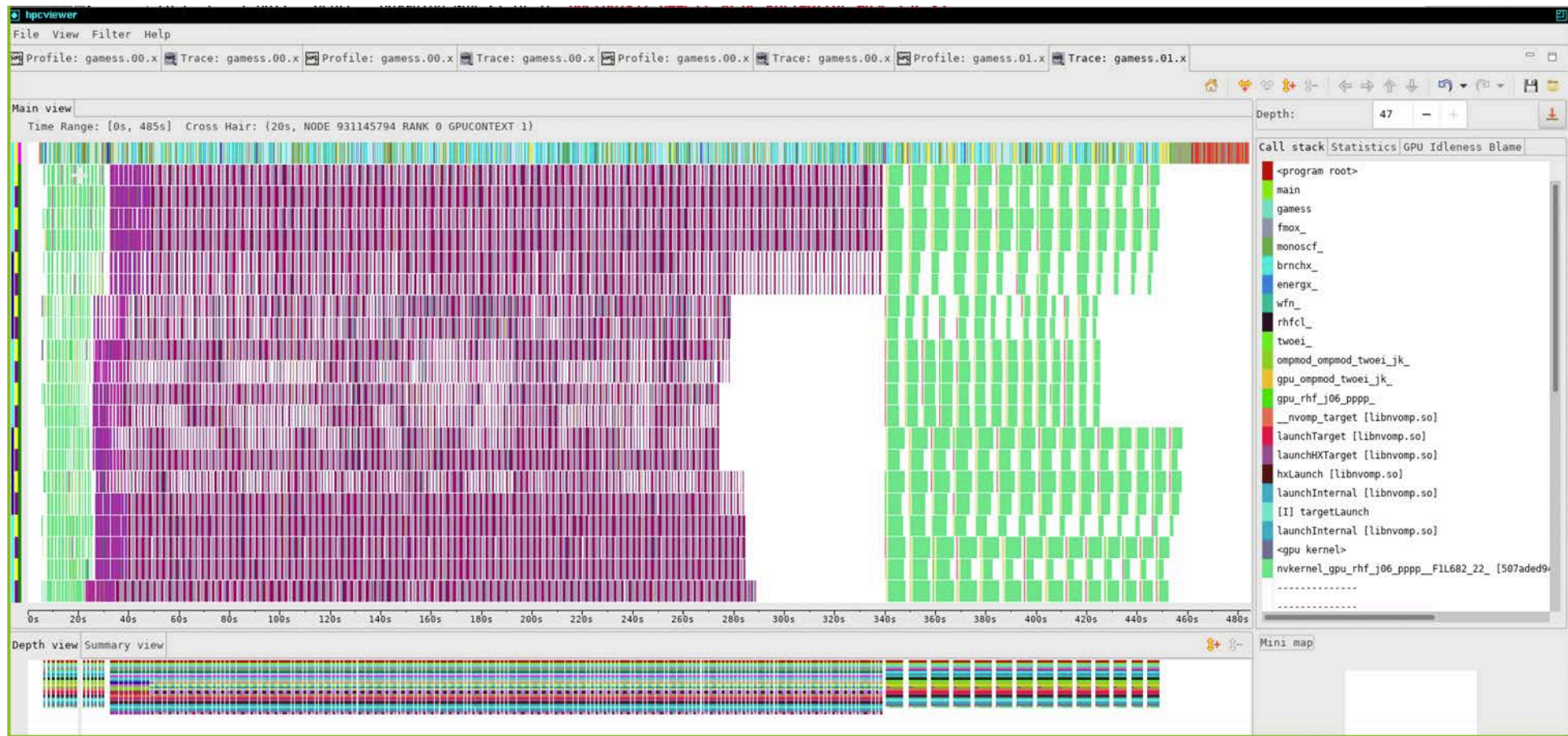
Time-centric Analysis: GAMESS 5 nodes, 40 ranks, 20 GPUs on Perlmutter



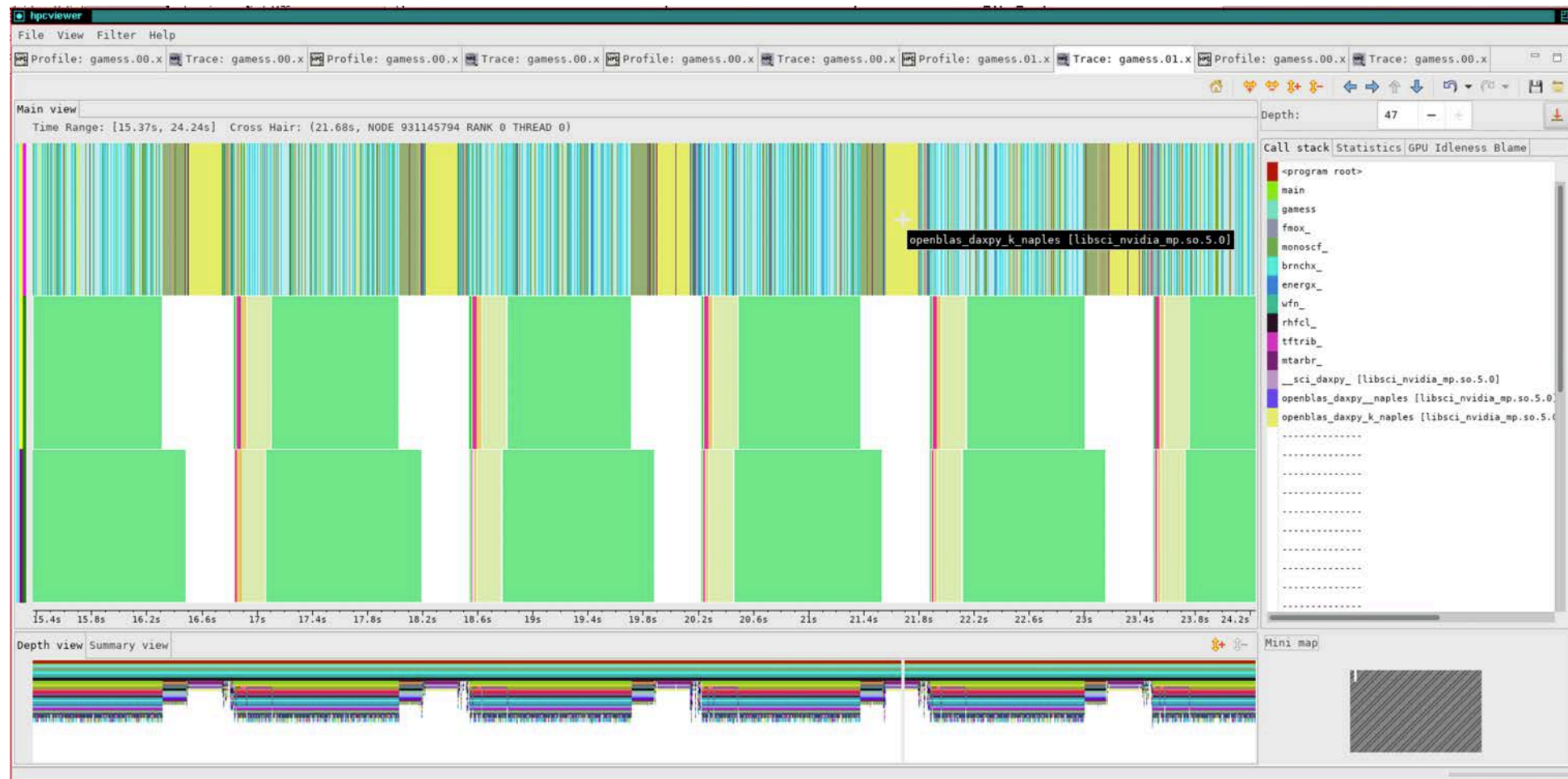
Time-centric Analysis: GAMESS 5 nodes, 40 ranks, 20 GPUs on Perlmutter



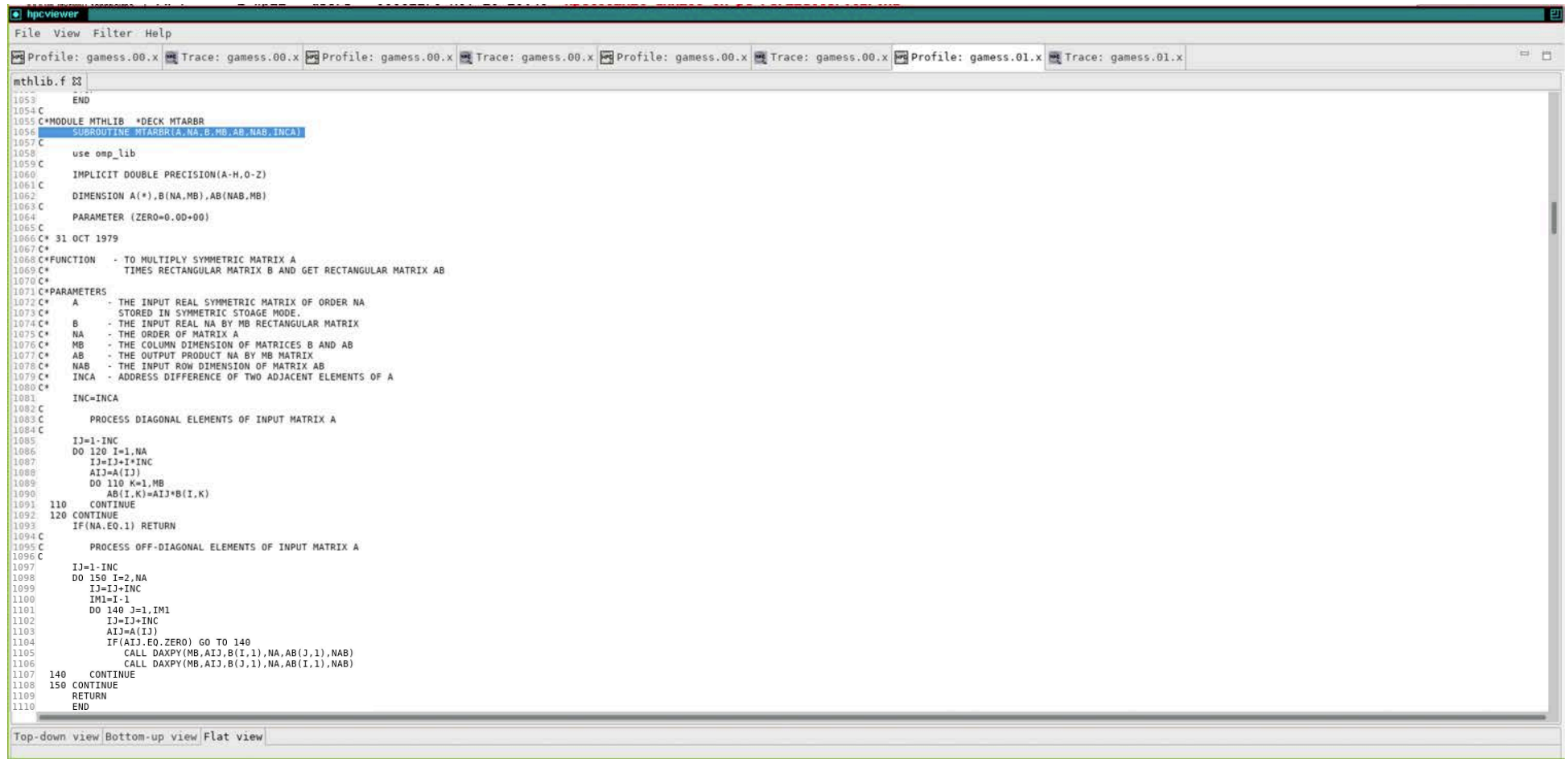
Time-centric Analysis: GAMESS 5 nodes, 40 ranks, 20 GPUs on Perlmutter



Time-centric Analysis: GAMESS 5 nodes, 40 ranks, 20 GPUs on Perlmutter



Time-centric Analysis: GAMESS 5 nodes, 40 ranks, 20 GPUs on Perlmutter

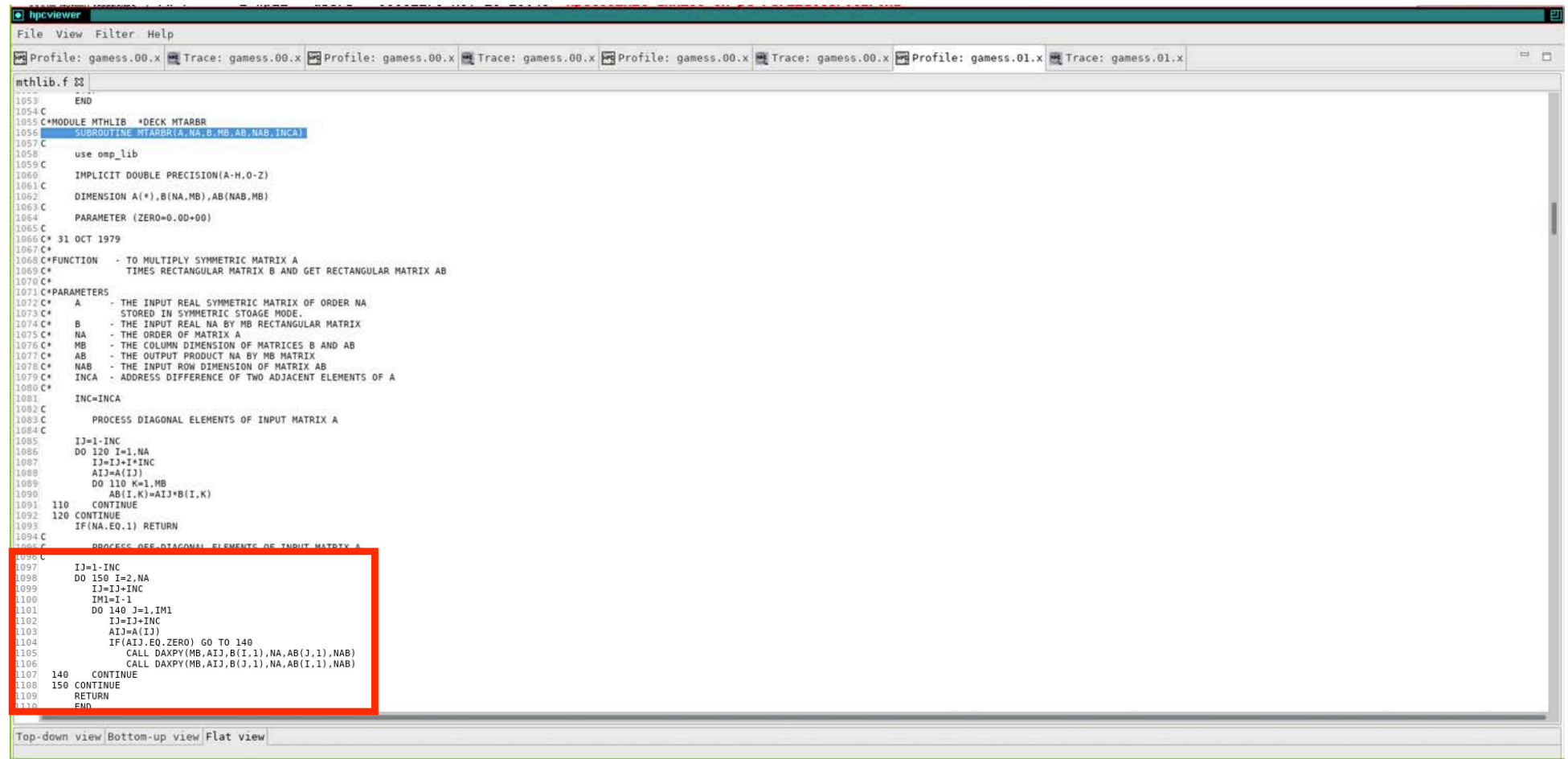


```
hpcviewer
File View Filter Help
Profile: gamess.00.x Trace: gamess.00.x Profile: gamess.00.x Trace: gamess.00.x Profile: gamess.00.x Trace: gamess.00.x Profile: gamess.01.x Trace: gamess.01.x

mthlib.f
1053 END
1054 C
1055 C*MODULE MTHLIB *DECK MTARBR
1056 SUBROUTINE MTARBR(A,NA,B,MB,AB,NAB,INCA)
1057 C
1058 use omp_lib
1059 C
1060 IMPLICIT DOUBLE PRECISION(A-H,O-Z)
1061 C
1062 DIMENSION A(*),B(NA,MB),AB(NAB,MB)
1063 C
1064 PARAMETER (ZERO=0.0D+00)
1065 C
1066 C* 31 OCT 1979
1067 C*
1068 C*FUNCTION - TO MULTIPLY SYMMETRIC MATRIX A
1069 C* TIMES RECTANGULAR MATRIX B AND GET RECTANGULAR MATRIX AB
1070 C*
1071 C*PARAMETERS
1072 C* A - THE INPUT REAL SYMMETRIC MATRIX OF ORDER NA
1073 C* STORED IN SYMMETRIC STORAGE MODE.
1074 C* B - THE INPUT REAL NA BY MB RECTANGULAR MATRIX
1075 C* NA - THE ORDER OF MATRIX A
1076 C* MB - THE COLUMN DIMENSION OF MATRICES B AND AB
1077 C* AB - THE OUTPUT PRODUCT NA BY MB MATRIX
1078 C* NAB - THE INPUT ROW DIMENSION OF MATRIX AB
1079 C* INCA - ADDRESS DIFFERENCE OF TWO ADJACENT ELEMENTS OF A
1080 C*
1081 INC=INCA
1082 C
1083 C PROCESS DIAGONAL ELEMENTS OF INPUT MATRIX A
1084 C
1085 IJ=1-INC
1086 DO 120 I=1,NA
1087 IJ=IJ+INC
1088 AIJ=A(IJ)
1089 DO 110 K=1,MB
1090 AB(I,K)=AIJ*B(I,K)
1091 110 CONTINUE
1092 120 CONTINUE
1093 IF(NA.EQ.1) RETURN
1094 C
1095 C PROCESS OFF-DIAGONAL ELEMENTS OF INPUT MATRIX A
1096 C
1097 IJ=1-INC
1098 DO 150 I=2,NA
1099 IJ=IJ+INC
1100 IM1=I-1
1101 DO 140 J=1,IM1
1102 IJ=IJ-INC
1103 AIJ=A(IJ)
1104 IF(AIJ.EQ.ZERO) GO TO 140
1105 CALL DAXPY(MB,AIJ,B(I,1),NA,AB(J,1),NAB)
1106 CALL DAXPY(MB,AIJ,B(J,1),NA,AB(I,1),NAB)
1107 140 CONTINUE
1108 150 CONTINUE
1109 RETURN
1110 END

Top-down view Bottom-up view Flat view
```

Time-centric Analysis: GAMESS 5 nodes, 40 ranks, 20 GPUs on Perlmutter

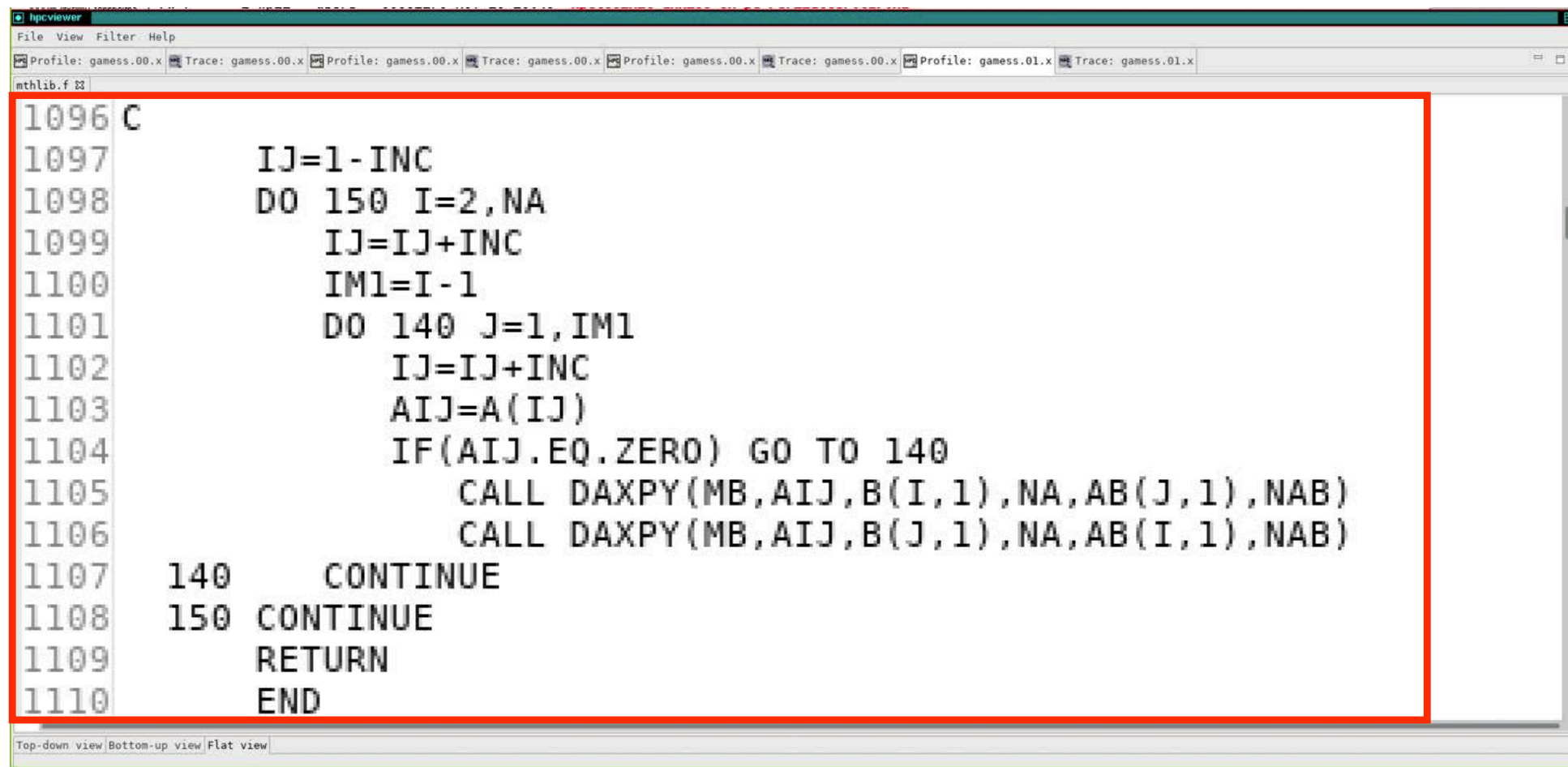


```
hpcviewer
File View Filter Help
Profile: gameess.00.x Trace: gameess.00.x Profile: gameess.00.x Trace: gameess.00.x Profile: gameess.01.x Trace: gameess.01.x

mthlib.f
1053 END
1054 C
1055 C*MODULE MTHLIB *DECK MTARBR
1056 SUBROUTINE MTARBR(A,NA,B,MB,AB,NAB,INCA)
1057 C
1058 use omp_lib
1059 C
1060 IMPLICIT DOUBLE PRECISION(A-H,O-Z)
1061 C
1062 DIMENSION A(*),B(NA,MB),AB(NAB,MB)
1063 C
1064 PARAMETER (ZERO=0.0D+00)
1065 C
1066 C* 31 OCT 1979
1067 C*
1068 C*FUNCTION - TO MULTIPLY SYMMETRIC MATRIX A
1069 C* TIMES RECTANGULAR MATRIX B AND GET RECTANGULAR MATRIX AB
1070 C*
1071 C*PARAMETERS
1072 C* A - THE INPUT REAL SYMMETRIC MATRIX OF ORDER NA
1073 C* STORED IN SYMMETRIC STORAGE MODE.
1074 C* B - THE INPUT REAL NA BY MB RECTANGULAR MATRIX
1075 C* NA - THE ORDER OF MATRIX A
1076 C* MB - THE COLUMN DIMENSION OF MATRICES B AND AB
1077 C* AB - THE OUTPUT PRODUCT NA BY MB MATRIX
1078 C* NAB - THE INPUT ROW DIMENSION OF MATRIX AB
1079 C* INCA - ADDRESS DIFFERENCE OF TWO ADJACENT ELEMENTS OF A
1080 C*
1081 INC=INCA
1082 C
1083 C PROCESS DIAGONAL ELEMENTS OF INPUT MATRIX A
1084 C
1085 IJ=1-INC
1086 DO 120 I=1,NA
1087 IJ=IJ+INC
1088 AIJ=A(IJ)
1089 DO 110 K=1,MB
1090 AB(I,K)=AIJ*B(I,K)
1091 110 CONTINUE
1092 120 CONTINUE
1093 IF(NA.EQ.1) RETURN
1094 C
1095 C PROCESS OFF-DIAGONAL ELEMENTS OF INPUT MATRIX A
1096 C
1097 IJ=1-INC
1098 DO 150 I=2,NA
1099 IJ=IJ+INC
1100 IM1=I-1
1101 DO 140 J=1,IM1
1102 IJ=IJ+INC
1103 AIJ=A(IJ)
1104 IF(AIJ.EQ.ZERO) GO TO 140
1105 CALL DAXPY(MB,AIJ,B(I,1),NA,AB(J,1),NAB)
1106 CALL DAXPY(MB,AIJ,B(J,1),NA,AB(I,1),NAB)
1107 140 CONTINUE
1108 150 CONTINUE
1109 RETURN
1110 END

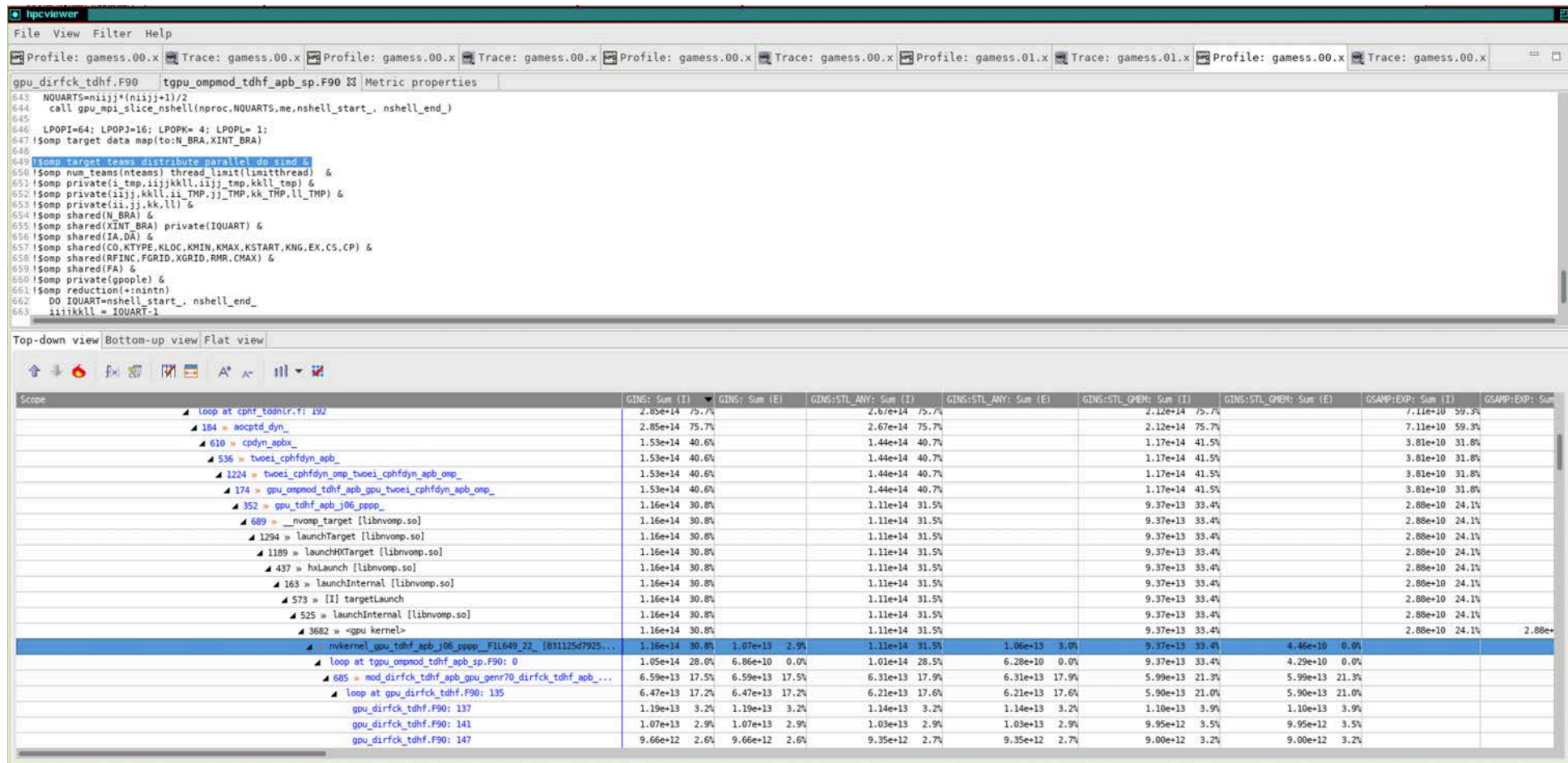
Top-down view Bottom-up view Flat view
```

Time-centric Analysis: GAMESS 5 nodes, 40 ranks, 20 GPUs on Perlmutter

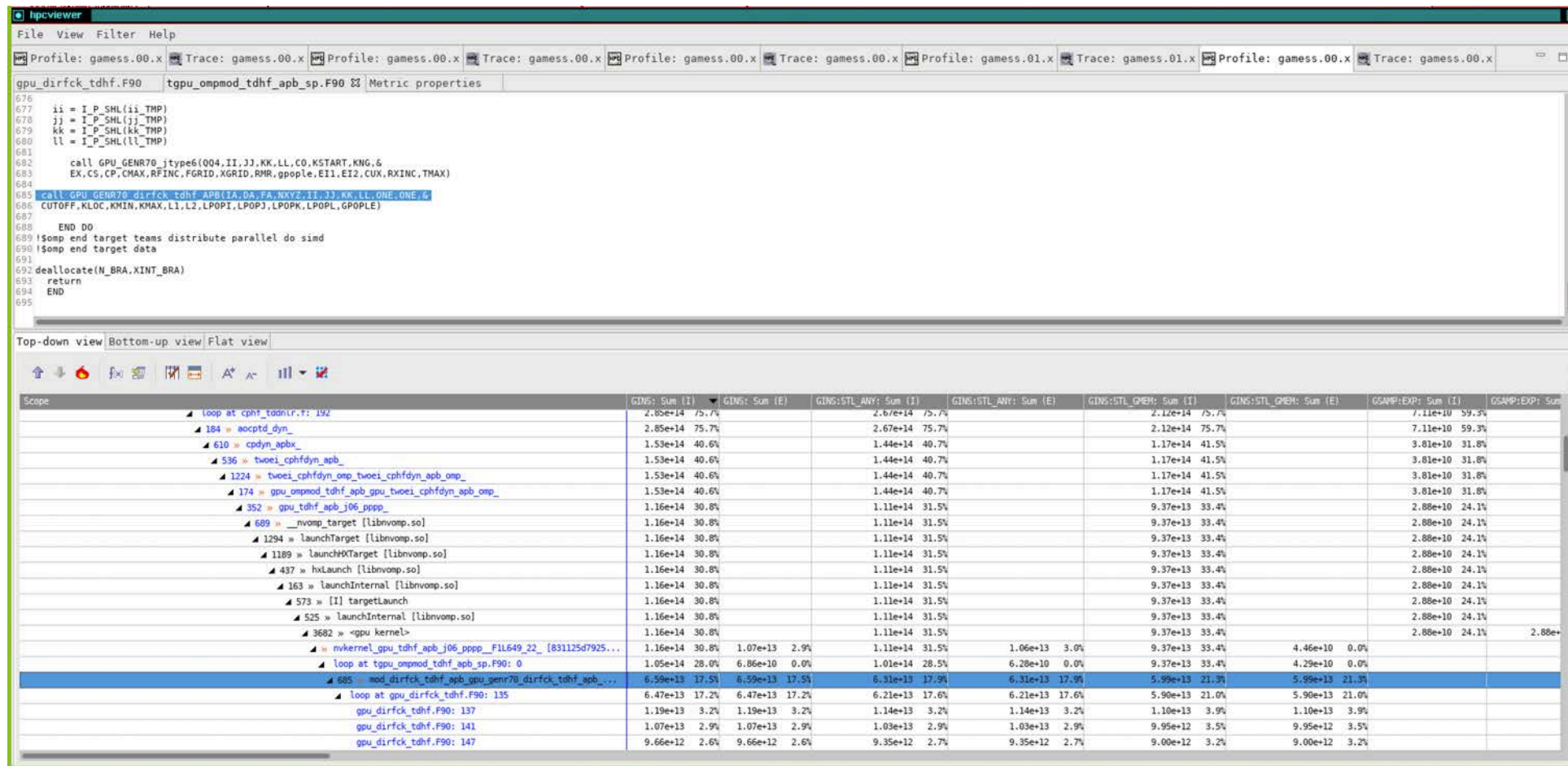


```
hpcviewer
File View Filter Help
Profile: gameess.00.x Trace: gameess.00.x Profile: gameess.00.x Trace: gameess.00.x Profile: gameess.00.x Trace: gameess.00.x Profile: gameess.01.x Trace: gameess.01.x
mthlib.f
1096 C
1097     IJ=1-INC
1098     DO 150 I=2,NA
1099         IJ=IJ+INC
1100         IM1=I-1
1101         DO 140 J=1,IM1
1102             IJ=IJ+INC
1103             AIJ=A(IJ)
1104             IF(AIJ.EQ.ZERO) GO TO 140
1105                 CALL DAXPY(MB,AIJ,B(I,1),NA,AB(J,1),NAB)
1106                 CALL DAXPY(MB,AIJ,B(J,1),NA,AB(I,1),NAB)
1107 140     CONTINUE
1108 150 CONTINUE
1109     RETURN
1110     END
Top-down view Bottom-up view Flat view
```

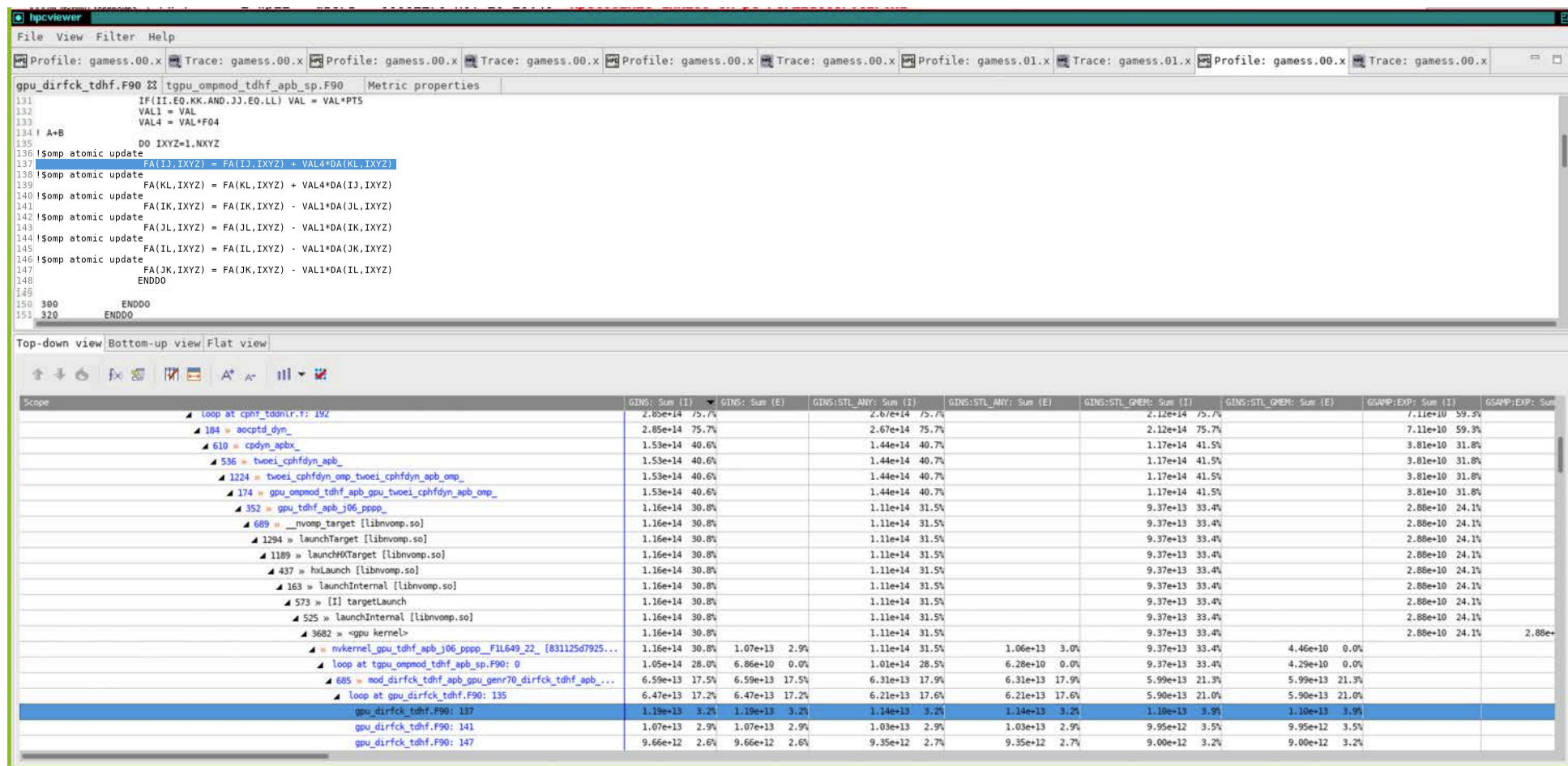

Time-centric Analysis: GAMESS 5 nodes, 40 ranks, 20 GPUs on Perlmutter



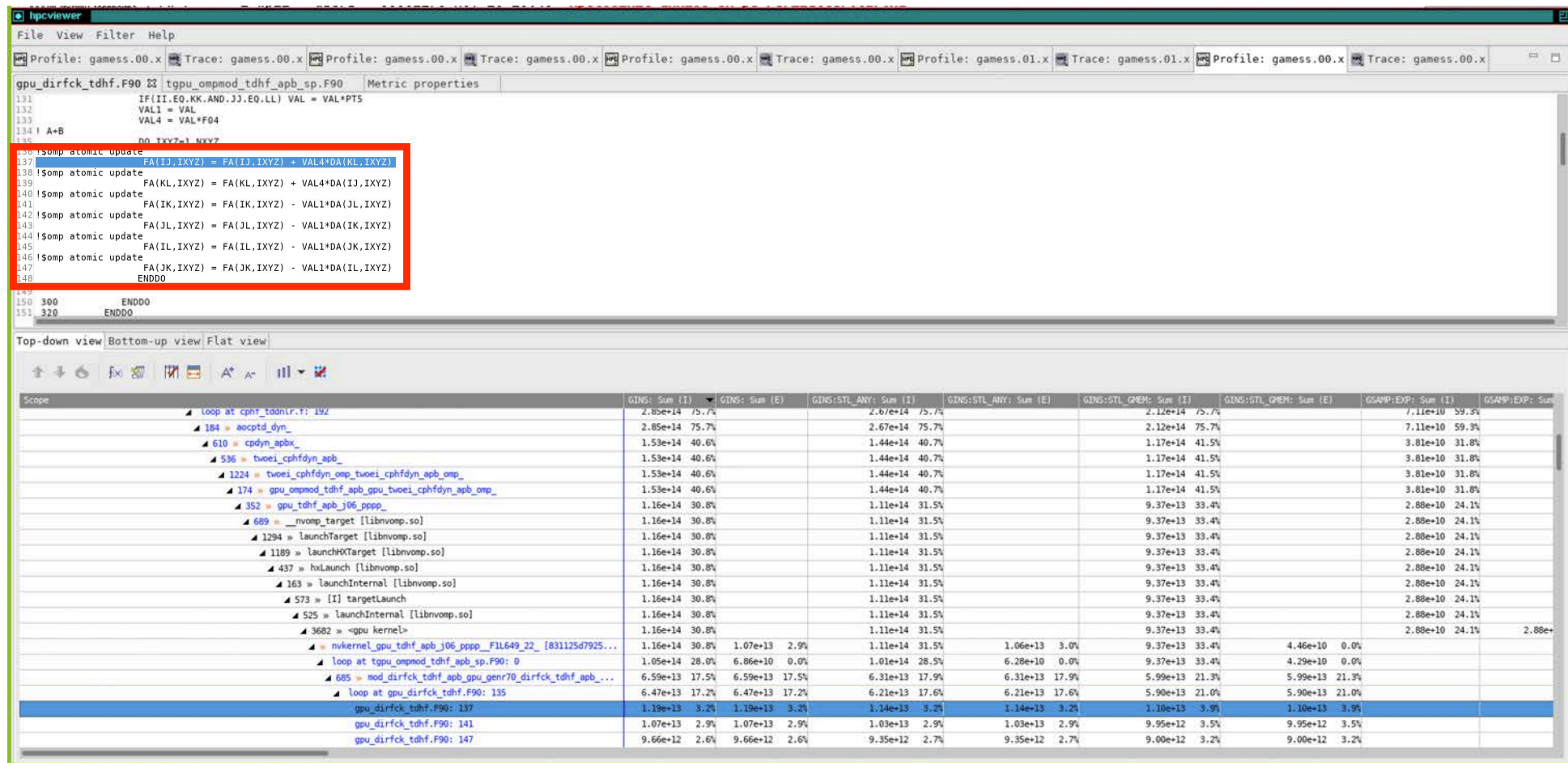
Time-centric Analysis: GAMESS 5 nodes, 40 ranks, 20 GPUs on Perlmutter



Time-centric Analysis: GAMESS 5 nodes, 40 ranks, 20 GPUs on Perlmutter



Time-centric Analysis: GAMESS 5 nodes, 40 ranks, 20 GPUs on Perlmutter



Time-centric Analysis: GAMESS 5 nodes, 40 ranks, 20 GPUs on Perlmutter

The screenshot displays the hpcviewer application. The top pane shows the source code of the `mod_dirfck_tdhf_apb` module, specifically the `loop at gpu_dirfck_tdhf.F90: 135`. The code includes several atomic update operations for the `FA` array. The bottom pane shows a performance table with columns for rank, time, and percentage. The table is filtered to show data for the `gpu_dirfck_tdhf.F90: 135` loop.

```

131 IF((II.EQ.KK.AND.JJ.EQ.LL) VAL = VAL*PTS
132 VAL1 = VAL
133 VAL4 = VAL*F04
134 A+B
135 DO IXYZ=1,NXYZ
136 !$omp atomic update
137     FA(IJ,IXYZ) = FA(IJ,IXYZ) + VAL4*DA(KL,IXYZ)
138 !$omp atomic update
139     FA(KL,IXYZ) = FA(KL,IXYZ) + VAL4*DA(IJ,IXYZ)
140 !$omp atomic update
141     FA(IK,IXYZ) = FA(IK,IXYZ) - VAL1*DA(JL,IXYZ)
142 !$omp atomic update
143     FA(JL,IXYZ) = FA(JL,IXYZ) - VAL1*DA(IK,IXYZ)
144 !$omp atomic update
145     FA(IL,IXYZ) = FA(IL,IXYZ) - VAL1*DA(JK,IXYZ)
146 !$omp atomic update
147     FA(JK,IXYZ) = FA(JK,IXYZ) - VAL1*DA(IL,IXYZ)
148 ENDDO
    
```

Rank	Time	Percentage
685	6.59e+13	17.5%
686	6.59e+13	17.5%
687	6.31e+13	17.9%
688	6.31e+13	17.9%
689	5.99e+13	21.3%
690	5.99e+13	21.3%
691	6.47e+13	17.2%
692	6.47e+13	17.2%
693	6.21e+13	17.6%
694	6.21e+13	17.6%
695	5.90e+13	21.0%
696	5.90e+13	21.0%
697	1.19e+13	3.2%
698	1.19e+13	3.2%
699	1.14e+13	3.2%
700	1.14e+13	3.2%
701	1.10e+13	3.9%
702	1.10e+13	3.9%
703	1.07e+13	2.9%
704	1.07e+13	2.9%
705	1.03e+13	2.9%
706	1.03e+13	2.9%
707	9.95e+12	3.5%
708	9.95e+12	3.5%
709	9.66e+12	2.6%
710	9.66e+12	2.6%
711	9.35e+12	2.7%
712	9.35e+12	2.7%
713	9.00e+12	3.2%
714	9.00e+12	3.2%

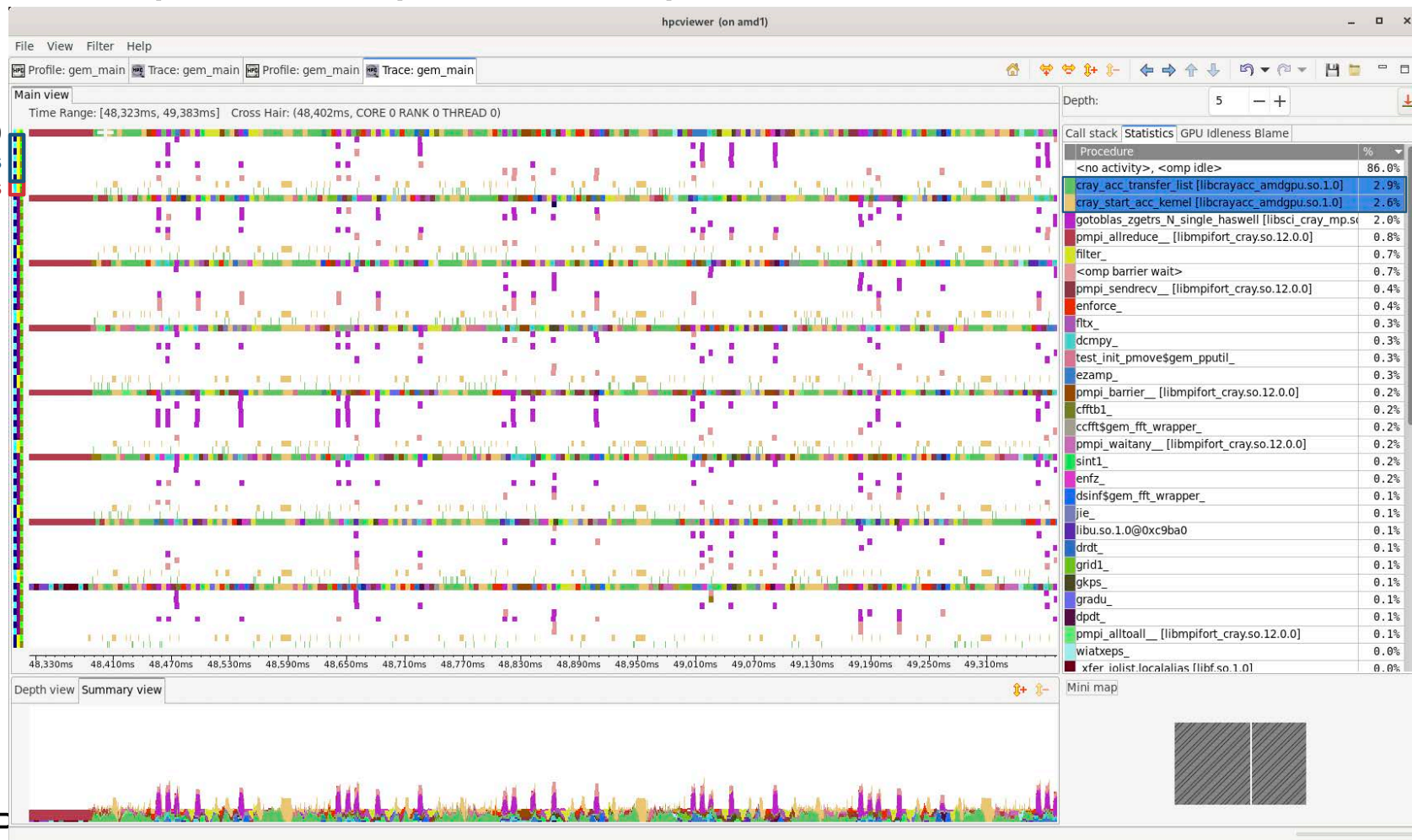
Case Study: GEM (Gyrokinetic Turbulence Code)

- GEM: a comprehensive electromagnetic delta-f particle-in-cell code that includes the full dynamics of gyrokinetic ions and drift-kinetic electrons
 - Developed by University of Colorado at Boulder, part of ECP WDMApp project
- Code is written in Fortran 90 + MPI + OpenACC, with ongoing porting efforts to OpenMP target offload (https://dl.acm.org/doi/abs/10.1007/978-3-030-97759-7_7)
- Tested platforms: Perlmutter, Crusher, and Frontier using Cray compiler
 - Frontier: 16 nodes, 8 MPI ranks per node, 4 OpenMP threads per rank, 1 GPU per rank, 2 GPU streams per GPU device

Frontier	Wall-clock Time	Speedup
Without GPU offloading	290.88s	1 (base)
Naive GPU offloading	41.80s	6.96
Optimized GPU offloading	39.52s	7.36

First attempt: not all parallel loops should be offloaded

Rank 0 Thread 0
OpenMP threads
GPU streams



Too much data
movement
between CPU
& GPU

First attempt: not all parallel loops should be offloaded

hpcviewer (on amd1)

File View Filter Help

Profile: gem_main Trace: gem_main Profile: gem_main Trace: gem_main

Metric properties gem_pputil.f90 X gem_main.f90

```

58 CONTAINS
59 #ifndef OLD PMOVE
60 SUBROUTINE test_init_pmove(xp, np, lz, ierr)
61 !
62 REAL, DIMENSION(:), INTENT(in) :: xp !z position of particles
63 INTEGER, INTENT(in) :: np !total number of particles
64 REAL, INTENT(in) :: lz !total length of z direction
65 INTEGER, INTENT(out) :: ierr
66

```

Top-down view Bottom-up view Flat view

% GCOPY = 100 x GXCOPY / GPUOP

Scope	% GCOPY (I)	GPUOP (sec): Sum (I)	GPUOP (sec): Sum (E)	GKER (sec): Sum (I)	GKER (sec): Sum (E)	GXCOPY (sec): Sum (I)	GXCOPY (sec): Sum (E)
Experiment Aggregate Metrics	24.78 %	4.61e+01 100.0%	4.61e+01 100.0%	3.47e+01 100.0%	3.47e+01 100.0%	1.14e+01 100.0%	1.14e+01 100.0%
cray_acc_transfer_list [libcrayacc_amdgpu.so.1.0]	100.00 %	1.14e+01 24.8%	1.14e+01 24.8%			1.14e+01 100.0%	1.14e+01 100.0%
load_	100.00 %	4.22e-02 0.1%				4.22e-02 0.4%	
loader_wrapper_	100.00 %	1.32e-01 0.3%				1.32e-01 1.2%	
ldel_	100.00 %	8.97e-02 0.2%				8.97e-02 0.8%	
initialize_	100.00 %	5.54e-03 0.0%				5.54e-03 0.0%	
init_	100.00 %	5.54e-03 0.0%				5.54e-03 0.0%	
test_init_pmove\$gem_pputil_	97.11 %	2.18e+00 4.7%		6.31e-02 0.2%		2.12e+00 18.6%	
test_pmove\$gem_pputil_	45.53 %	6.38e+00 13.8%		3.48e+00 10.0%		2.91e+00 25.4%	
ppush_	40.47 %	3.50e+00 7.6%		2.08e+00 6.0%		1.42e+00 12.4%	
weatxeps_	35.89 %	1.59e-01 0.3%		1.02e-01 0.3%		5.70e-02 0.5%	
jpar0_	33.46 %	4.45e+00 9.7%		2.96e+00 8.5%		1.49e+00 13.0%	
ampere_	33.46 %	4.45e+00 9.7%		2.96e+00 8.5%		1.49e+00 13.0%	
grid1_	32.54 %	2.82e+00 6.1%		1.90e+00 5.5%		9.18e-01 8.0%	
accumulate_	31.98 %	3.07e+00 6.7%		2.08e+00 6.0%		9.80e-01 8.6%	
cpush_	31.05 %	4.60e+00 10.0%		3.17e+00 9.1%		1.43e+00 12.5%	
wiatxeps_	27.72 %	2.73e-01 0.6%		1.97e-01 0.6%		7.56e-02 0.7%	
pint_	27.71 %	7.01e+00 15.2%		5.07e+00 14.6%		1.94e+00 17.0%	
setw_	25.61 %	2.45e-01 0.5%		1.82e-01 0.5%		6.28e-02 0.6%	
push_wrapper_	24.87 %	2.71e+01 58.7%		2.03e+01 58.7%		6.73e+00 59.0%	
gem_main_	24.78 %	4.61e+01 100.0%		3.47e+01 100.0%		1.14e+01 100.0%	
split_weight_	17.81 %	1.09e+01 23.7%		8.98e+00 25.9%		1.95e+00 17.1%	
jie_	17.81 %	1.09e+01 23.7%		8.98e+00 25.9%		1.95e+00 17.1%	
cint_	16.26 %	1.20e+01 25.9%		1.00e+01 28.9%		1.94e+00 17.0%	

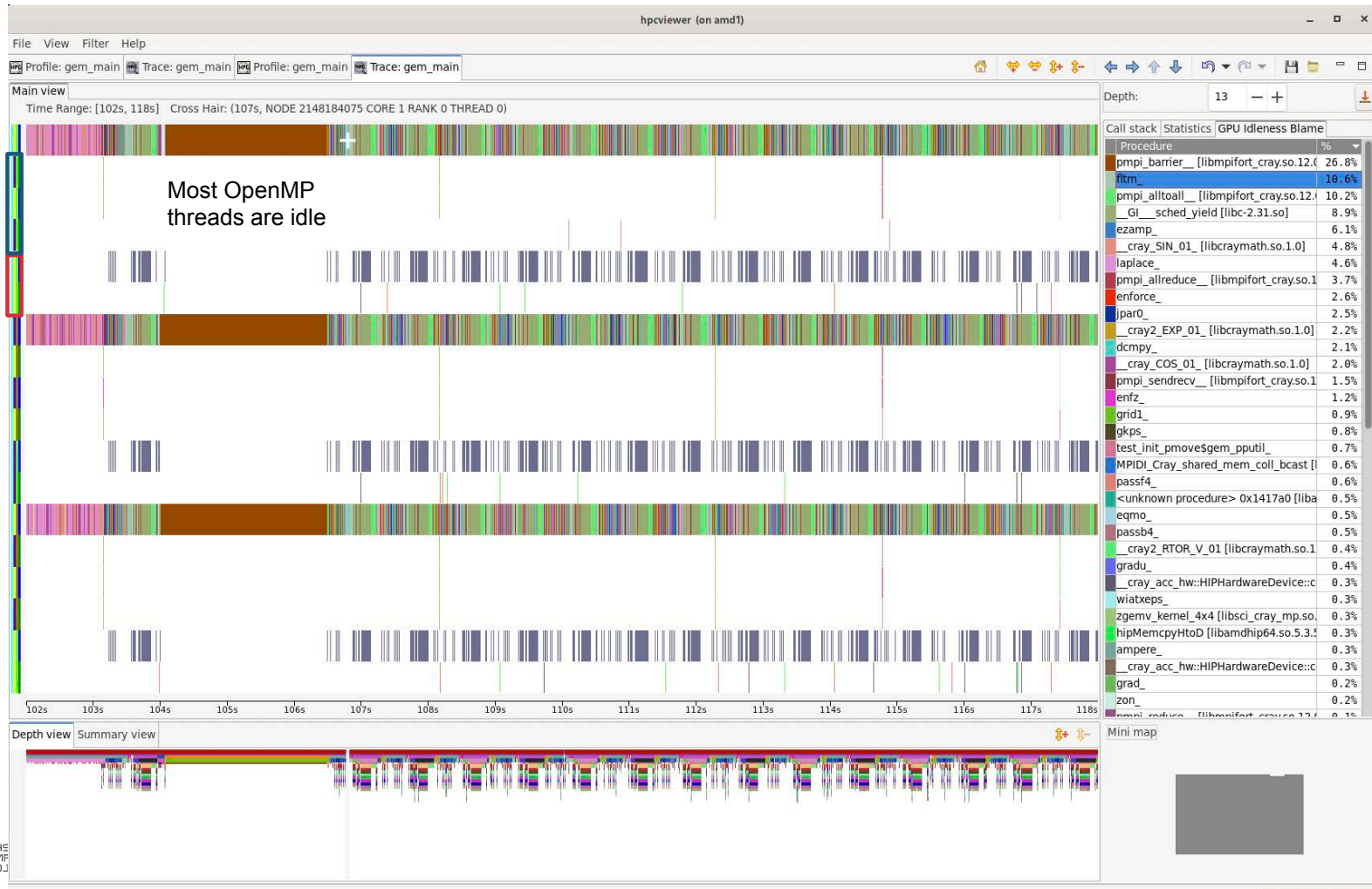
Procedures
test_init_pmove
and test_pmove
have high data
movement compared
to GPU computation

Use CPU threads to reduce GPU idleness

Rank 0 Thread 0

OpenMP threads

GPU streams



10.6% of GPU idle occurs when the main CPU thread executes `f1tm_` procedure.

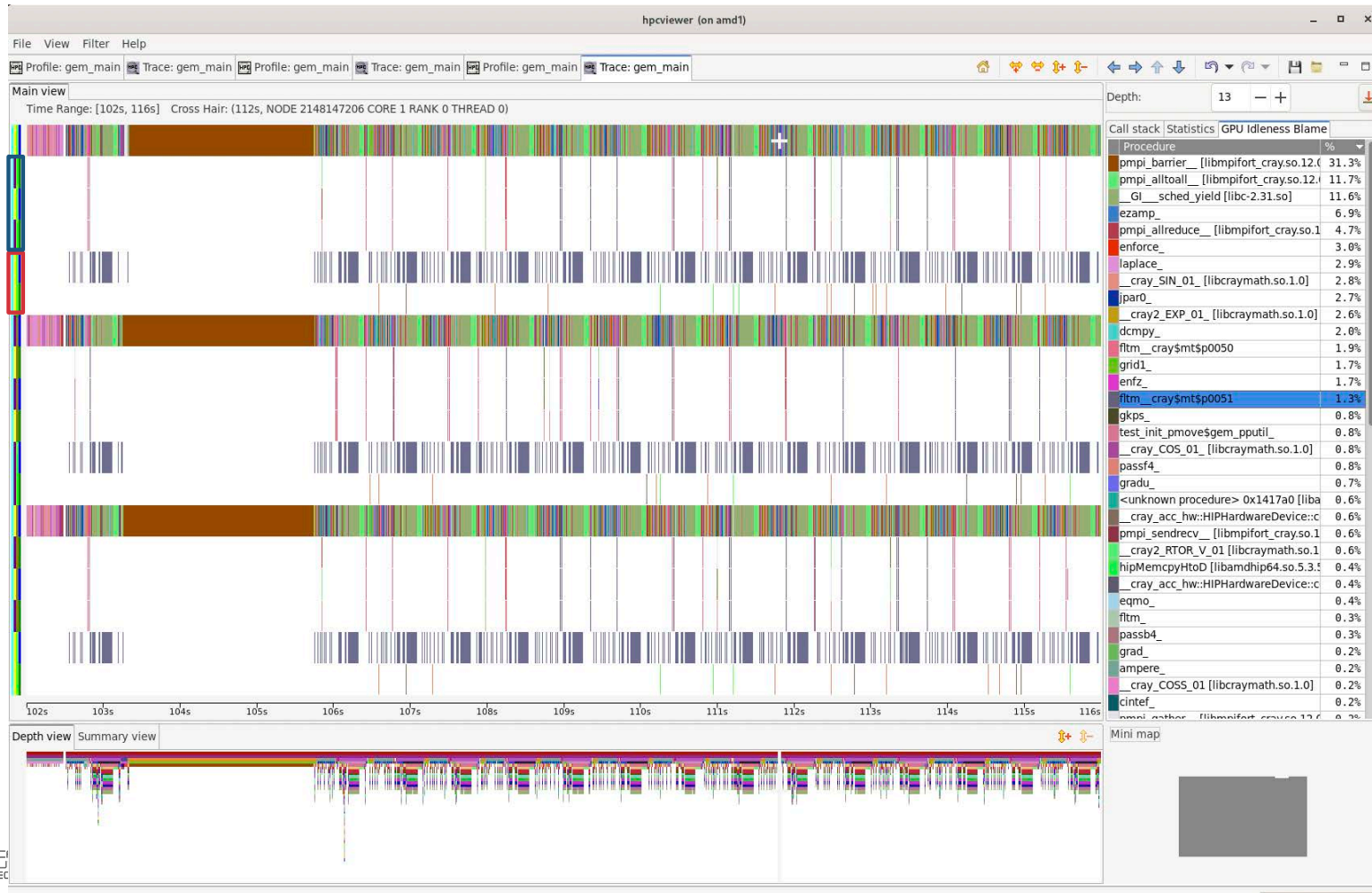
Parallelizing this procedure should reduce GPU idleness.

Final step: parallelizing `fltm_` procedure to reduce GPU idleness

Rank 0 Thread 0

OpenMP threads

GPU streams



HPCToolkit Status on GPUs

- NVIDIA
 - heterogeneous profiles
 - GPU instruction-level execution and stalls using PC sampling
 - traces
- AMD
 - heterogeneous profiles
 - no GPU instruction-level measurements within kernels
 - **measure OpenMP offloading using OMPT interface**
 - hardware counters to measure kernels
 - traces
- Intel
 - heterogeneous profiles
 - GPU instruction-level measurements with instrumentation; heuristic latency attribution to instructions
 - **measure OpenMP offloading using OMPT interface**
 - traces

Ongoing Work

- Enhancing measurement to identify root causes of scalability losses
 - identify measurement of delays caused by GPU and communication
- Developing comprehensive support for NVTX/ROCTX/Caliper/Kokkos Labels
- Support for instruction-level measurement and attribution on AMD and Intel GPUs
- Improving the scalability of hpcprof-mpi
 - avoid unnecessary serialization of I/O
- Developing new GUI support for analysis of remote data
- Adding a Python-based interface for analysis of performance results
 - developing a Python API to support arbitrary queries and analysis of profiles and traces
 - developing a tool that presents high-level performance reports
 - exploring automated analysis to identify notable features in executions
 - e.g. load imbalance, trace line equivalence classes