Identifying Top CPU Consuming COBOL Programs

Igor Cohen

Reid Copeland

Cindy Guo

Version Date: 19/08/2020

Contents

Introduction and motivation 1
Identifying peak usage times 2
Identifying top CPU consuming job-program pairs from peak usage times 3
Identifying top CPU consuming modules from job-program pairs 6
Identifying top CPU consuming COBOL CSECTs from modules 10
Summary 12
List of resources 13

Introduction and motivation

This document is for IBM Enterprise COBOL for z/OS and IBM Automatic Binary Optimizer (ABO) for z/OS clients to help identify their top CPU consuming COBOL programs.

Although your applications may have thousands of COBOL programs, many likely consume very little CPU. Optimizing all your programs using ABO or migrating all of them to the latest compiler can be expensive and time consuming. By targeting the most performance critical sections of your application code for optimization and migration, you can obtain most of the performance improvements of the latest optimizer and compiler technologies but with a much smaller effort.

This document provides a step by step process that starts from a system wide view of peak usage times and drills down to the exact COBOL programs that would deliver the most value from performance improvements.

Identifying peak usage times

Tools such as IBM z Batch Network Analyzer (zBNA) tool, IBM Sub-Capacity Reporting Tool (SCRT) and the z/OS Resource Measurement Facility (RMF) can be used to identify times of the day or days of the month when CPU usage is the highest.

zBNA is a no charge, PC based productivity tool to visualize CPU usage reports from SMF records. This document uses zBNA as the example tool to identify times of peak CPU usage. Once the zBNA client is opened and all required SMF records have been loaded (see zBNA Users Guide for details), you can display the rolling 4-hour average graph by clicking **Graph** > **Display Graph: Rolling 4 Hr Avg**.

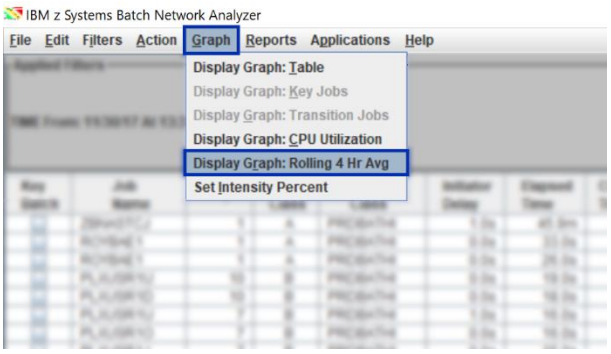


Figure 1a) Screenshot of zBNA showing the Graph tab

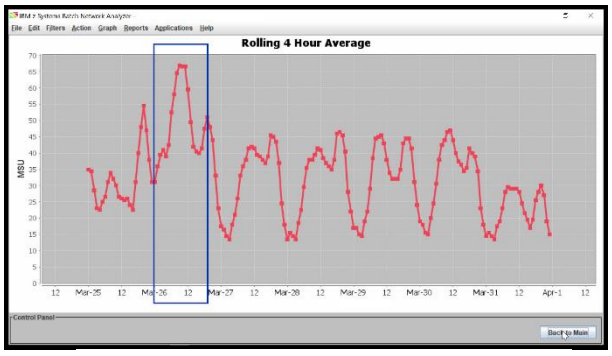


Figure 1b) Rolling 4 Hour Average for March

In this example, the peak usage is on March 26th.

Next, use zBNA "**Filters** > **Set Table Filters...**" to zoom into the day and find the peak usage interval.

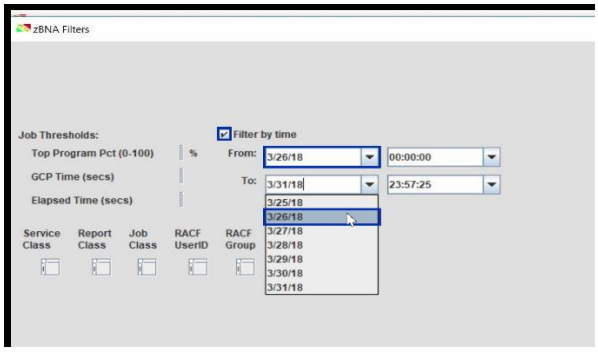


Figure 2a) Screenshot of zBNA showing the Filters page

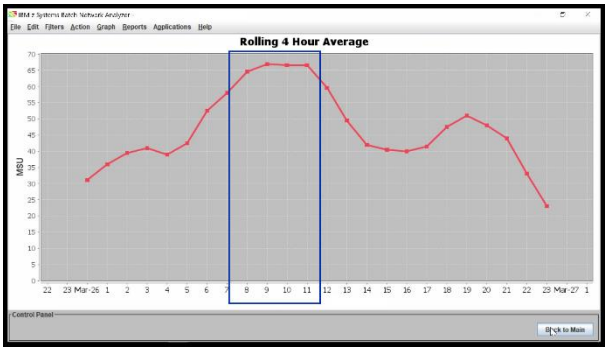


Figure 2b) Rolling 4 Hour Average on March 26th

In this example, the peak usage interval is from 7am to 11am.

Identifying top CPU consuming job-program pairs from peak usage times

With the peak CPU usage time now identified the top CPU consuming job-program pairs can be listed and extracted from zBNA.

First, add Step Level Records by clicking **File** > **Add Selected Step Level Records...**. Then use zBNA to save the CPU usage report as a comma-separated values (CSV) file with jobs and steps information by selecting **File** > **Save as CSV** > **Save as CSV, Jobs & Steps ...**.

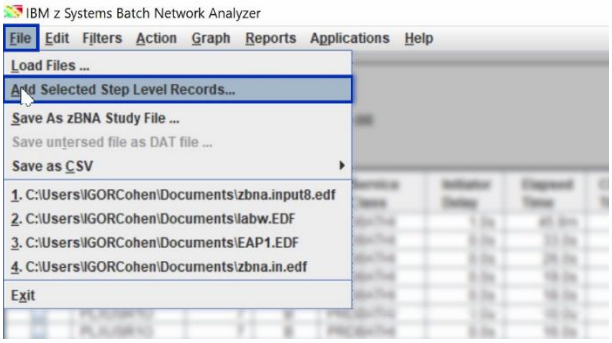


Figure 3a) Screenshot of zBNA showing the File tab

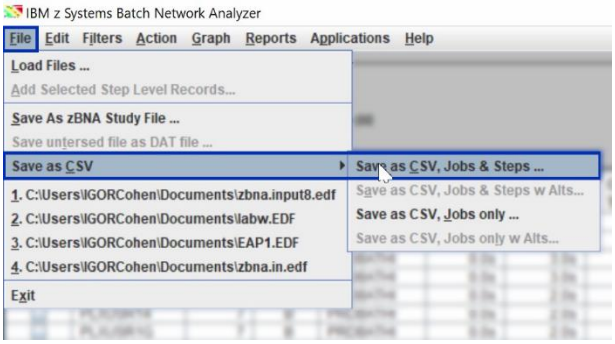


Figure 3b) Screenshot of zBNA showing the File tab

Next, generate the candidate job-program pairs and their cumulative CPU time by editing the CSV file.

> **Tip:** For users of zBNA earlier than 2.1, we have developed a tool to help you generate a job list from the CSV file. For more information, please go to https://www.ibm.com/support/pages/node/742651
>
> The zBNA 2.1 'Top programs' report can be used to help you generate a job list from the CSV file directly

1. In the CSV file you will see columns labelled "Job Name" and "Program Name".

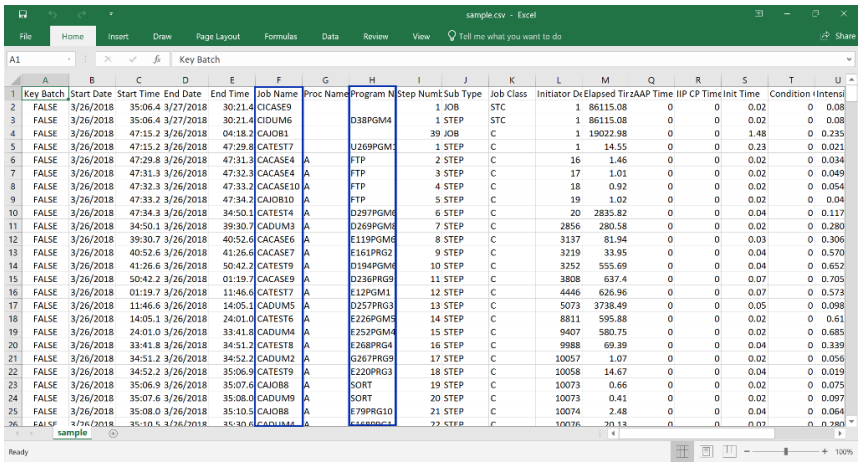


Figure 4 Jobs and Steps information shown in CSV format

Not all jobs listed here contain COBOL. You can remove known non-COBOL programs, such as, FTP, IEBCOPY, SORT, etc., tasks in job class STC, and TSO clients in job class TSU. The specific job class names may be different if you have customized them from the IBM defaults.

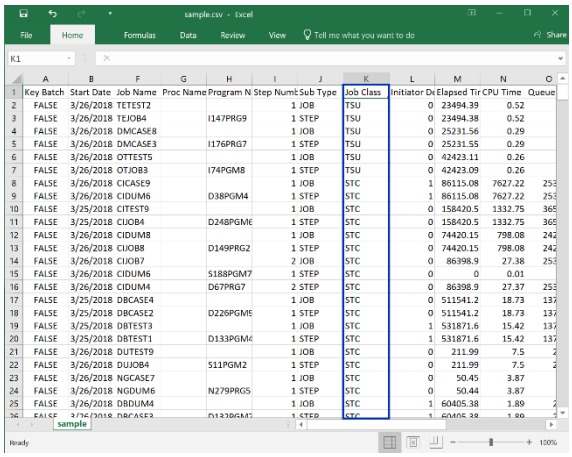


Figure 5a) Jobs in Jobs Class TSU and STC are highlighted

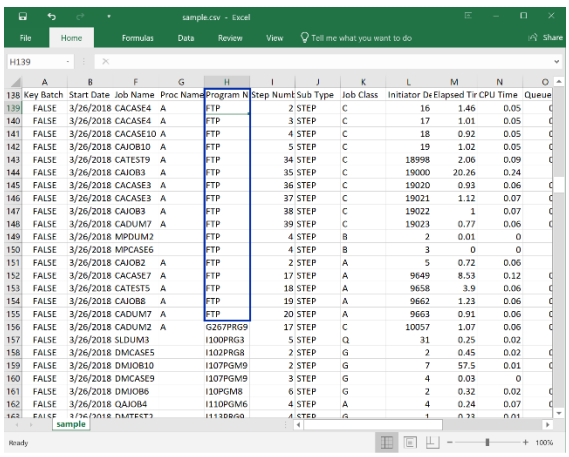
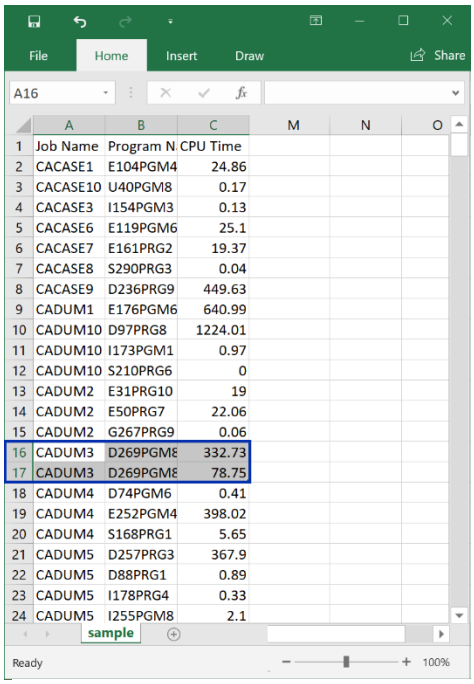


Figure 5b) Programs with name FTP are highlighted

2. Remove all columns except "Job Name", "Program Name" and "CPU Time". For any repeated job-program name pairs, the CPU Time should be added together.



	A	B	C
1	Job Name	Program N	CPU Time
2	CACASE1	E104PGM4	24.86
3	CACASE10	U40PGM8	0.17
4	CACASE3	I154PGM3	0.13
5	CACASE6	E119PGM6	25.1
6	CACASE7	E161PRG2	19.37
7	CACASE8	S290PRG3	0.04
8	CACASE9	D236PRG9	449.63
9	CADUM1	E176PGM6	640.99
10	CADUM10	D97PRG8	1224.01
11	CADUM10	I173PGM1	0.97
12	CADUM10	S210PRG6	0
13	CADUM2	E31PRG10	19
14	CADUM2	E50PRG7	22.06
15	CADUM2	G267PRG9	0.06
16	CADUM3	D269PGM8	332.73
17	CADUM3	D269PGM8	78.75
18	CADUM4	D74PGM6	0.41
19	CADUM4	E252PGM4	398.02
20	CADUM4	S168PRG1	5.65
21	CADUM5	D257PRG3	367.9
22	CADUM5	D88PRG1	0.89
23	CADUM5	I178PRG4	0.33
24	CADUM5	I255PGM8	2.1

Figure 6 Job-program pair with repeated entry

3. Sort the job-program list by “CPU Time” column from largest to smallest to see the top contributors.

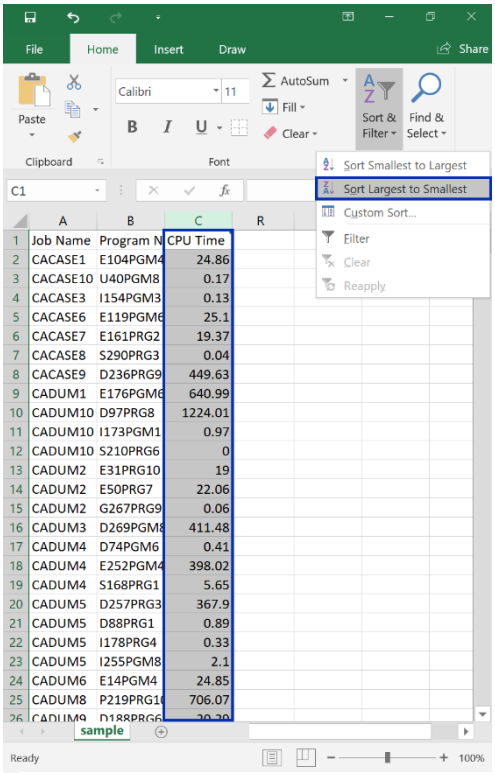


Figure 7a) Sort the final list based on CPU time

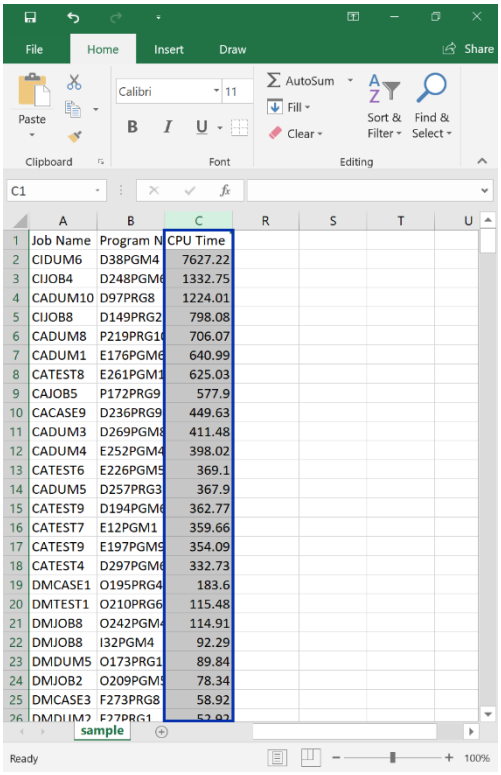


Figure 7b) Final job-program pairs with most CPU time

Now you have your top contributor job-program list sorted from most to least CPU time used. This list, however, may contain thousands of programs. To focus on the job-program pairs that contribute the bulk of the CPU time, compute the cumulative percentage of CPU time taken at each point in the list.

In the example above the first 15 candidates take 90.65% of the CPU time for all chosen candidates and 41.80% of the CPU time across all job-program pairs (that is, including all the non-COBOL programs previously filtered out). Therefore, it is sufficient to focus on the top 15 candidates from this point on.

Generate a CSV file containing the top job-program pairs you have chosen and copy it to your mainframe as the input for the next step.

```
CIDUM6,D38PGM4
CIJOB4,D248PGM6
CADUM10,D97PRG8
CIJOB8,D149PRG2
CADUM8,P219PRG1
CADUM1,E176PGM6
CATEST8,E261PGM1
CAJOB5,P172PRG9
...
```

Figure 8 Example file to be copied using FTP to your mainframe

Identifying top CPU consuming modules from job-program pairs

Now it's time to find the top CPU consuming modules used by the selected job-program pairs. This step requires a performance analysis tool such as IBM Application Performance Analyzer for z/OS (APA). If you do not have access to APA or any performance analysis tools, ABO includes a tool, called Run Time Instrumentation (RTI) Profiler, that can help determine the modules that are executed the most while your application is running. For more information on RTI Profiler, please go to ABO's User Guide.

The following steps outline how you can identify all modules used by the top contributing jobs and programs using APA.

> **Tip:** We have provided REXX script samples to help with this process for those who use APA. For more information, please go to https://www.ibm.com/support/pages/node/742651

1. Create an APA batch measurement request using the top contributor job-program list and run this batch job before the peak usage time you have previously identified.
   - Use `JOBNAME` parameter to specify the job name and `STEP` parameter to specify the corresponding program name of a single job-program pair.
   - Use the `RUNAGAIN` parameter for continuous monitoring of repeated job-programs.

```
//CAZBATCH EXEC PGM=CAZBATCH,PARM='STCID=CAZ0'
//STEPLIB DD DISP=SHR,DSN=APA.APA14G.SCAZAUTH
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
NEW JOBNAME=CIDUM6 STEP=(,D38PGM4) ACTIVE=NO
     RUNAGAIN=(180,FOR=99) RUNTOEOS=YES ;
NEW JOBNAME=CIJOB4 STEP=(,D248PGM6) ACTIVE=NO
     RUNAGAIN=(180,FOR=99) RUNTOEOS=YES ;
NEW JOBNAME=CADUM10 STEP=(,D97PRG8) ACTIVE=NO
     RUNAGAIN=(180,FOR=99) RUNTOEOS=YES ;
NEW JOBNAME=CIJOB8 STEP=(,D149PRG2) ACTIVE=NO
     RUNAGAIN=(180,FOR=99) RUNTOEOS=YES ;
NEW JOBNAME=CADUM8 STEP=(,P219PRG1) ACTIVE=NO
     RUNAGAIN=(180,FOR=99) RUNTOEOS=YES ;
NEW JOBNAME=CADUM1 STEP=(,E176PGM6) ACTIVE=NO
     RUNAGAIN=(180,FOR=99) RUNTOEOS=YES ;
NEW JOBNAME=CATEST8 STEP=(,E261PGM1) ACTIVE=NO
     RUNAGAIN=(180,FOR=99) RUNTOEOS=YES ;
NEW JOBNAME=CAJOB5 STEP=(,P172PRG9) ACTIVE=NO
     RUNAGAIN=(180,FOR=99) RUNTOEOS=YES ;
/*
```

2. Create two APA batch jobs to print the following reports and run these jobs after the peak usage time you have previously identified:
   - Batch job 1: To print the S03 "Load Module Summary" report
   - Batch job 2: To print the C08 "CPU Usage Referred Attribution" and C02 "CPU Usage by Module" reports

S03 report creation job sample is shown below.

```
//CAZBATCH EXEC PGM=CAZBATCH,PARM='STCID=CAZ0'
//CAZPRNT1 EXEC PGM=CAZPRINT
//STEPLIB DD DISP=SHR,DSN=APA.APA14G.SCAZAUTH
//SFILE01 DD DISP=SHR,DSN=APA.YOURID.R0708.CIDUM6.SF
//SFILE02 DD DISP=SHR,DSN=APA.YOURID.R0709.CIJOB4.SF
//SFILE03 DD DISP=SHR,DSN=APA.YOURID.R0710.CADUM10.SF
//SFILE04 DD DISP=SHR,DSN=APA.YOURID.R0711.CIJOB8.SF
//SFILE05 DD DISP=SHR,DSN=APA.YOURID.R0712.CADUM8.SF
//SFILE06 DD DISP=SHR,DSN=APA.YOURID.R0713.CADUM1.SF
//SFILE07 DD DISP=SHR,DSN=APA.YOURID.R0714.CATEST8.SF
//SFILE08 DD DISP=SHR,DSN=APA.YOURID.R0715.CAJOB5.SF
//CAZLOG DD SYSOUT=*
//PRINT1 DD DSN=&SYSUID..APA.PRINTS03, UNIT=SYSDA,  /* print output */
// SPACE=(CYL,(5,5),RLSE),DCB=(LRECL=255,RECFM=VB,BLKSIZE=0),DISP=(,CATLG)
//CAZCTL DD *
 PROFILE 01 INPUT=SFILE01
 SECTION S03 PROFILE=01
 PROFILE 02 INPUT=SFILE02
 SECTION S03 PROFILE=02
 PROFILE 03 INPUT=SFILE03
 SECTION S03 PROFILE=03
 ...
 PROFILE 07 INPUT=SFILE07
 SECTION S03 PROFILE=07
 PROFILE 08 INPUT=SFILE08
 SECTION S03 PROFILE=08
 PRINT DDNAME=PRINT1
```

C08/C02 combined report creation job sample is shown below.

```
//CAZBATCH EXEC PGM=CAZBATCH,PARM='STCID=CAZ0'
//CAZPRNT1 EXEC PGM=CAZPRINT
//STEPLIB DD DISP=SHR,DSN=APA.APA14G.SCAZAUTH
//SFILE01 DD DISP=SHR,DSN=APA.YOURID.R0708.CIDUM6.SF
//SFILE02 DD DISP=SHR,DSN=APA.YOURID.R0709.CIJOB4.SF
//SFILE03 DD DISP=SHR,DSN=APA.YOURID.R0710.CADUM10.SF
//SFILE04 DD DISP=SHR,DSN=APA.YOURID.R0711.CIJOB8.SF
//SFILE05 DD DISP=SHR,DSN=APA.YOURID.R0712.CADUM8.SF
//SFILE06 DD DISP=SHR,DSN=APA.YOURID.R0713.CADUM1.SF
//SFILE07 DD DISP=SHR,DSN=APA.YOURID.R0714.CATEST8.SF
//SFILE08 DD DISP=SHR,DSN=APA.YOURID.R0715.CAJOB5.SF
//CAZLOG DD SYSOUT=*
//PRINT1 DD DSN=&SYSUID..APA.PRINTC08, UNIT=SYSDA,  /* print output */
// SPACE=(CYL,(5,5),RLSE),DCB=(LRECL=255,RECFM=VB,BLKSIZE=0),DISP=(,CATLG)
//CAZCTL DD *
 PROFILE 01 INPUT=SFILE01
 SECTION C08 PROFILE=01
 SECTION C02 PROFILE=01
 PROFILE 02 INPUT=SFILE02
 SECTION C08 PROFILE=02
 SECTION C02 PROFILE=02
 ...
 PROFILE 10 INPUT=SFILE10
 SECTION C08 PROFILE=10
 SECTION C02 PROFILE=10

 PRINT DDNAME=PRINT1
```

The S03 report shows the name and location for all statically and dynamically called programs and displays each module's location within a Load Library. Below is a sample report for job CADUM8.

```
S03: Load Module Summary (00712/CADUM8)

Module   Locn  Address Count Size(bytes)  Attributes  DDName    Load Library

CALLEE1   JPA  268DAC48   1       5,048              STEPLIB   HLQ.SAMPLE.COBPGM.INLOAD
C612GHP1  JPA  268FAC64   1      68,048              STEPLIB   HLQ.SAMPLE.COBPGM.INLOAD
CALLEE2   JPA  268DCC48   1       5,048              STEPLIB   HLQ.SAMPLE.COBPGM.INLOAD
PGNSWJ13  JPA  268FBC64   1      38,048              STEPLIB   HLQ.SAMPLE.COBPGM.INLOAD
CALLEE3   JPA  268DEC48   1       5,048              STEPLIB   HLQ.SAMPLE.COBPGM.INLOAD
CEEBINIT  JPA  00007D88   1      45,688   RU RN      STEPLIB   TSCTEST.CEEZ220.SCEERUN
CEEPLPKA  JPA  26611780   1   2,181,248   RU RN      STEPLIB   TSCTEST.CEEZ220.SCEERUN
IEAVEWAT  NUC  00FF2500   1       4,556
IEWFETCH  NUC  0165CE20   1       7,632
IGWCCA00  PLPA 0A57B000   1     177,824                        SYS1.LPALIB
IGZCEV5   JPA  26826AB0   1      17,744   RU RN      STEPLIB   TSCTEST.CEEZ220.SCEERUN
IGZCPAC   JPA  2682B8E0   1     448,288   RU RN      STEPLIB   TSCTEST.CEEZ220.SCEERUN
I419PRGB  JPA  266008B8   1       5,960              STEPLIB   HLQ.SAMPLE.COBPGM.INLOAD
```

The C08 report displays each module's relative contribution to overall job CPU.

```
C08: CPU Usage Referred Attribution (00712/CADUM8)

Name      Description                Percent of CPU Time * 10.00%  +/-3.9%
                                          *....1....2....3....4....5....6....7....8....9....*
 C612GHP1  Application Program       30.67 ===============
 > C612GHP1  CSECT in C612GHP1       30.67 ===============
   > 00074C    Attribution Offset    30.67 ===============
               in C612GHP1
     > IGZCPAC   COBPACK             30.67 ***************
       > IGZCXDI   Double            29.29 **************
                   precision
                   division
       > IGZCXMU   Double             1.38 *
                   precision
                   multiplication


 CALLEE1   Application Program       23.15 ============
> CALLEE1   CSECT in CALLEE1         23.15 ============
  > 00050C    Attribution Offset    23.15 ============
              in CALLEE1
```

The C02 report displays a module's relative contribution to overall job CPU time as in C08. But, a module with low system services activity can be in both C02 and C08, or in C02 report only.

```
C02: CPU Usage by Module (00712/CADUM8)

 Name      Description        Percent of CPU Time * 10.00%  +/-6.3%
                                  *....1....2....3....4....5....6....7....8....9....*
 C612GHP1  Application      100.00 **************************************************
           Program
 > C612GHP1  CSECT in       100.00 **************************************************
             C612GHP1
```

3. From the C02 and C08 reports for each job-program pair, take the line for each module with "`Application`" in its description and create the combined report shown below that lists the modules that potentially could be COBOL programs.

```
I419PRGB  Application Program    22.00 ===========
C612GHP1  Application Program    99.46 ========================================
PGNSWJ13  Application Program    30.67 ===============
CALLEE1   Application Program    23.15 ============
CALLEE3   Application Program    23.15 ============
CALLEE2   Application Program    23.00 ============
CALLEE6   Application Program    90.24 ========================================
CALLEE4   Application Program     2.43 =
CALLEE5   Application Program     5.62 ===
I210017A  Application Program     0.60


C612GHP1  Application           100.00 ************************************
I419PRGB  Application             7.40 ****
```

Modules that contribute very little to the overall time (e.g. CALLEE4 at 2.43% and I210017A at 0.60%) can be excluded if desired as collectively they take only about 3% of the overall time.

Next, cross-reference the module list above against the S03 location information to determine the fully qualified location to be scanned for COBOL CSECTs. For example, CALLEE5 is located at `HLQ.SAMPLE.COBPGM.INLOAD(CALLEE5)`.

Identifying top CPU consuming COBOL CSECTs from modules

The top CPU contributing modules may or may not contain COBOL CSECTs (compiled programs). There are several tools available to scan your modules to identify the COBOL CSECTs.

IBM File Manager for z/OS can be used to identify the COBOL CSECTs and the build compiler version in each module. The File Manager's "`View Load Module`" function can be used from ISPF on-line, TSO or REXX exec to display compiler information.

```
Address CSECT name        Type Size    Class      AMODE RMODE Date     Compiler 1
        *                 *            *          *     *     *        *
<---+-> <---+----10---+-> <--> <---+-> <---+----> <---> <---> <---+--> <---+----10---+----2----+-->
0000000 CALLEE5           SD   0000CC8 B_TEXT     MIN   ANY   2010.027 Enterpr.COBOL for z/OS V4R2
```

ABO contains a built-in SCAN=Y mode that can also display the CSECTs eligible for optimization. List the modules to be scanned as input to ABO using the `SCAN=Y` option as shown.

```
//SYSIN DD *
  SCAN=Y
  BOPT IN=HLQ.SAMPLE.COBPGM.INLOAD(I419PRGB)
  BOPT IN=HLQ.SAMPLE.COBPGM.INLOAD(PGNSWJ13)
  BOPT IN=HLQ.SAMPLE.COBPGM.INLOAD(CALLEE1)
  BOPT IN=HLQ.SAMPLE.COBPGM.INLOAD(CALLEE2)
  BOPT IN=HLQ.SAMPLE.COBPGM.INLOAD(CALLEE3)
  BOPT IN=HLQ.SAMPLE.COBPGM.INLOAD(C612GHP1)
  BOPT IN=HLQ.SAMPLE.COBPGM.INLOAD(CALLEE5)
  BOPT IN=HLQ.SAMPLE.COBPGM.INLOAD(CALLEE6)
```

ABO will display the eligible COBOL CSECTs in each module. In this example, module I419PRGB contains no COBOL CSECTs that are eligible for optimization, so it can be removed from the final candidate list of modules to optimize with ABO.

```
5697-AB1 IBM Automatic Binary Optimizer for z/OS 1.3.0

======== Monday Oct 20 2018 ========
12:12:39 Optimizer build level: tr_r17_binopt_20180924_141188 (Sept 24 2018 14:05:42)
12:12:39 Processing HLQ.SAMPLE.COBPGM.INLOAD, member CALLEE5
   Language ID Records:
      id 5655S7100 v42 m00 2010027 resident CALLEE5
         Enterprise COBOL V4: start=0x0, length=3.20 (kBytes)
            Signature information bytes:
               a0487d4c 20000000 00880100 00000040
               08000000 000000 00008004 1400
12:12:39 Processing HLQ.SAMPLE.COBPGM.INLOAD, member I419PRGB
   Language ID Records:
      id 569623400  v01 m06 2018005 resident I419PRGB
         BOZ4101: No applicable COBOL code section found
...
12:12:40 Exiting with return code: 0
```

With the non-COBOL module `I419PRGB` removed the final list of the top contributing COBOL modules is below.

```
HLQ.SAMPLE.COBPGM.INLOAD(PGNSWJ13)
HLQ.SAMPLE.COBPGM.INLOAD(CALLEE1)
HLQ.SAMPLE.COBPGM.INLOAD(CALLEE2)
HLQ.SAMPLE.COBPGM.INLOAD(CALLEE3)
HLQ.SAMPLE.COBPGM.INLOAD(C612GHP1)
HLQ.SAMPLE.COBPGM.INLOAD(CALLEE5)
HLQ.SAMPLE.COBPGM.INLOAD(CALLEE6)
```

This list, in the format shown below, is ready to be used as input to ABO to optimize these modules to the specified `OUT=DD:SYSBOUT` location.

```
//SYSIN DD *
  ARCH=12
  BOPT IN=HLQ.SAMPLE.COBPGM.INLOAD(PGNSWJ13) OUT=DD:SYSBOUT
  BOPT IN=HLQ.SAMPLE.COBPGM.INLOAD(CALLEE1)  OUT=DD:SYSBOUT
  BOPT IN=HLQ.SAMPLE.COBPGM.INLOAD(CALLEE2)  OUT=DD:SYSBOUT
  BOPT IN=HLQ.SAMPLE.COBPGM.INLOAD(CALLEE3)  OUT=DD:SYSBOUT
  BOPT IN=HLQ.SAMPLE.COBPGM.INLOAD(C612GHP1) OUT=DD:SYSBOUT
  BOPT IN=HLQ.SAMPLE.COBPGM.INLOAD(CALLEE5)  OUT=DD:SYSBOUT
  BOPT IN=HLQ.SAMPLE.COBPGM.INLOAD(CALLEE6)  OUT=DD:SYSBOUT
```

If instead you are looking for to migrate your build compiler to the latest version of Enterprise COBOL, locate the corresponding source files for the COBOL CSECTs. Many source code management tools allow you to map modules and CSECTs back to the corresponding source file locations.

Summary

Migrating to the latest version of the Enterprise COBOL compiler or optimizing existing COBOL modules with ABO is an effective way to reduce CPU consumption, lower operating costs and shorten batch windows. Your applications might contain thousands of program modules; therefore, it is advantageous to target top CPU contributing modules for migration or optimization first. This document provides a step-by-step guide to help identify top CPU consuming COBOL modules in your applications. The process is broken down into four major steps: 1) identify peak usage time, 2) identify top CPU consuming job-program pairs, 3) identify top CPU consuming modules, and 4) identify COBOL CSECTs from these modules.

List of resources

How to Identify Top CPU Consuming COBOL Modules video:
https://ibm.biz/cobol-top-modules-video

IBM Z Identify Top COBOL Module Tools:
https://www.ibm.com/support/pages/node/742651

IBM Enterprise COBOL for z/OS:
https://www.ibm.com/products/cobol-compiler-zos

IBM Automatic Binary Optimizer for z/OS:
https://www.ibm.com/products/automatic-binary-optimizer-zos

IBM Application Performance Analyzer for z/OS (APA):
https://www.ibm.com/us-en/marketplace/application-performance-analyzer

IBM File Manager for z/OS:
https://www.ibm.com/us-en/marketplace/file-manager-for-zsystems

IBM System z Batch Network Analyzer (zBNA):
http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS5132