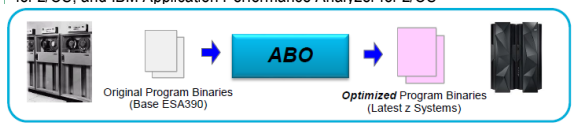
- The Automatic Binary Optimizer for z/OS V2.2‡ improves the performance of already compiled COBOL programs.
- The Optimizer does not require source code, source code migration, or performance options tuning.
  - It uses modern optimization technology to target z Systems, including z13, zEC12, and zBC12, to accelerate the performance of existing compiled COBOL applications.
- Leverages the latest advanced COBOL optimization technology to improve the performance of your already compiled COBOL programs — Reduces processing time and reduces execution costs.
- Generates code to target the latest deployment systems including the z13 processors
- Supports IBM Problem Determination Tools for z/OS including IBM Fault Analyzer for z/OS, IBM Debug Tool for z/OS, and IBM Application Performance Analyzer for z/OS

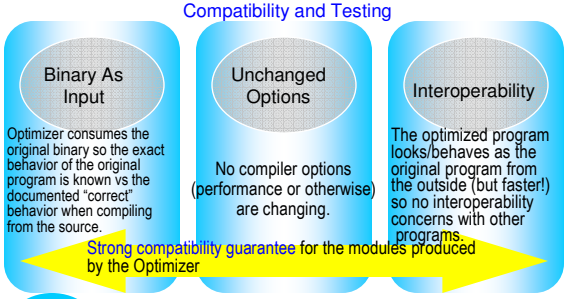


Optimizes directly from compiled COBOL programs and eliminates source level migrations, recompilations, and tuning performance options.

Use Case	ABO	Compiler
Significant Performance Improvement* *No Source, Migration or Options Tuning Required	✓	
Interoperability/Legacy Compatibility PDS support, pre-Enterprise COBOL etc.	✓	
Built in support for targeting multiple hardware levels at deployment	✓	
No need to downgrade ARCH setting to match DR* machine Original compiled program always available for DR	✓	
New COBOL development and new features		✓
Maintenance on existing COBOL programs		✓
Maximum Performance Improvement* *Source, Migration and Options Tuning Required		✓

* DR "Disaster Recovery" Machine : Down level machine used for emergency situations. Usually 1 or 2 revisions old so puts limits on Compiler ARCH setting (and performance improvements possible) based on this older level.

Compatibility and Testing



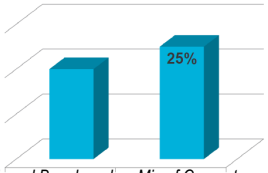
This product delivers the optimization technology to exploit the latest z Systems, which include z13 & zEC12/zBC12.

Version 1.1 decreases processing time and reduces MLC costs for running business critical applications.

Failure Scenario — If something goes wrong using the optimized module revert to using original program and call IBM

Testing — The recommended testing focus should be on performance and scalability (less on business logic)

Performance Internal Benchmark Suite and Early Customer Results Higher is better



- Early customer results show performance gains of **5- 21%** for a mix of v3 and v4 compiled input programs
- Performance gains will vary by application but expected to average 15%

Internal Benchmarks : Mix of Compute and I/O Bound Applications –z13
Higher is better : Optimizer gives a 25% Improvement

Note: Performance data contained above was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

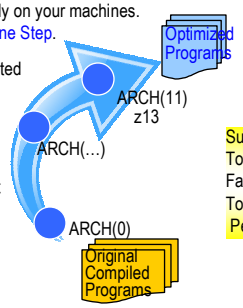
ABO or Latest COBOL Compiler?
Short answer : ***Both!***
They serve different but complementary purposes

Lets the user specify the dataset of the optimized modules.

Lets you generate code to target the latest deployment systems (z13, zEC12, and zBC12).

Using the Optimizer on COBOL v3 and v4 executables is a **25 year** jump forward in hardware technology and access to over **600 new hardware** instructions already on your machines.
ABO -Architecture Exploitation 0 to 11 In One Step.

- All Pre-V5 COBOL compiler releases generated only base ESA/390 level code
- The Optimizer, like COBOL V5, generates code up to z13 ARCH(11)



ABO leverages z/OS operating system support to automatically load optimized modules without requiring application JCL changes.

Supports IBM Problem Determination Tools for z/OS, which includes IBM Fault Analyzer for z/OS, IBM Debug Tool for z/OS, and IBM Application Performance Analyzer for z/OS.

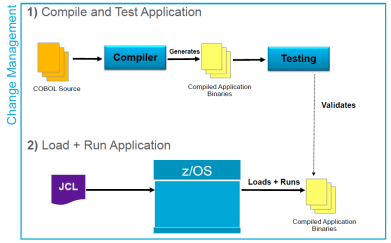
Use Senarios

Assumption: Current COBOL development and deployment processes are tightly integrated to change management tools and procedures for source control, tracking and auditing purposes

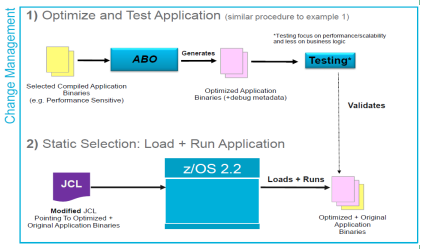
1)Current process using the compiler
2)Optimization process and static selection
3)Optimization process and dynamic selection
4)Optimization process using a hybrid approach

Summary: The development change management control, tracking and auditing processes, when using the Optimizer, are similar to current scenarios using the compiler.

Example 1:
Current Process – COBOL Development, Compilation, Test and Deployment

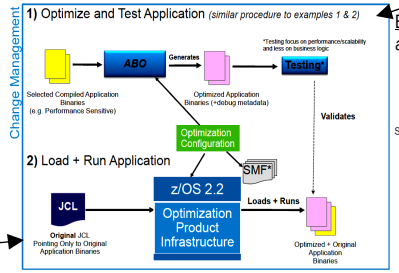


Example 2:
Binary Optimization Process Optimization, Test and ***Static*** Selection



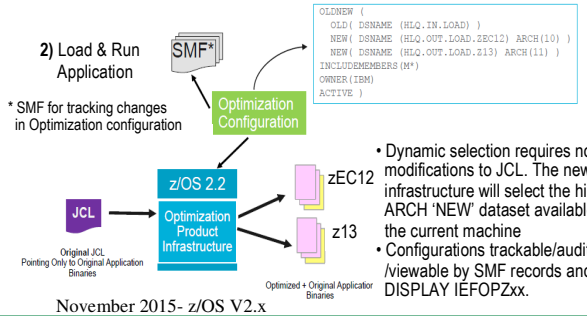
Example 3:
Binary Optimization Process Optimization, Test and ***Dynamic*** Selection

* SMF for tracking changes in optimization configuration



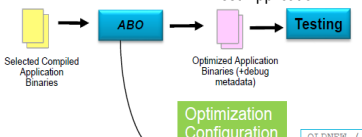
Example 3 In Depth: *Binary Optimization* Process **Optimization, Test and *Dynamic*** Selection (**targeting multiple hardware levels**)

```
...
//STEPLIB  DD DSN=$HLQ.IN.LOAD,DISP=SHR  ← e.g. target z13 and zEC12
...
```



- Dynamic selection requires no modifications to JCL. The new z/OS infrastructure will select the highest ARCH 'NEW' dataset available for the current machine
- Configurations trackable/auditable /viewable by SMF records and DISPLAY IEFOPZxx.

Example 3 In Depth: *Binary Optimization* Process **Optimization, Test and *Dynamic* Selection**



- Using the Optimizer in the dynamic selection case only requires "pointing" to active configuration by specifying IEFOPZ in JCL.

```
OLDNEW (
  OLD( DSNAME (HLQ.IN.LOAD) )
  NEW( DSNAME (HLQ.OUT.LOAD.ZEC12) ARCH(10) )
  NEW( DSNAME (HLQ.OUT.LOAD.Z13) ARCH(11) )
INCLUDEMEMBERS(M*)
OWNER(IBM)
INACTIVE )
```

```
//OPTIMIZE JOB
//OPT      EXEC PGM=BOZOPT,REGION=0M
//STEPLIB  DD DSN=$HLQ.BOZ110.SBOZMOD1,DISP=SHR
//SYSPRINT DD DSN=$HLQ.BOZOUT.LISTING,DISP=SHR
//SYSIN    DD *
  IEFOPZ
```

User Scenarios – Hybrid Approach

- There is no requirement when using dynamic selection to also use the Optimization Configuration to drive the optimization process itself.
- This "hybrid approach" uses the explicit new JCL from example 2 for the optimization, test and change management steps combined with the optimization configuration from example 3 for easier deployment across multiple hardware levels.

Example 4:
Binary Optimization Process –Hybrid Approach Specify binaries to optimize explicitly but combine with Dynamic Selection
1) Optimize and Test Application *(same procedure from example 2)*
2) Load & Run Application *(same procedure from example 3)*

‡ Anticipation to roll back to z/OS V2.1 (1Q'16)