Achieving Scalability in Rule Based Systems

George Williamson

Associate Systems Engineer

Union Pacific Railroad

Agenda

- Introduction
- Car Scheduling Overview
- Yard Block Assignment
- Implementation
- Achieving Scalability

Introduction

- George Williamson
- Assoc. Systems Engineer at Union Pacific Railroad
- Decision Technologies
- Internal Consultant for Rules-Based Development.
- A developer that builds software to satisfy real-world business needs.

Car Scheduling Overview

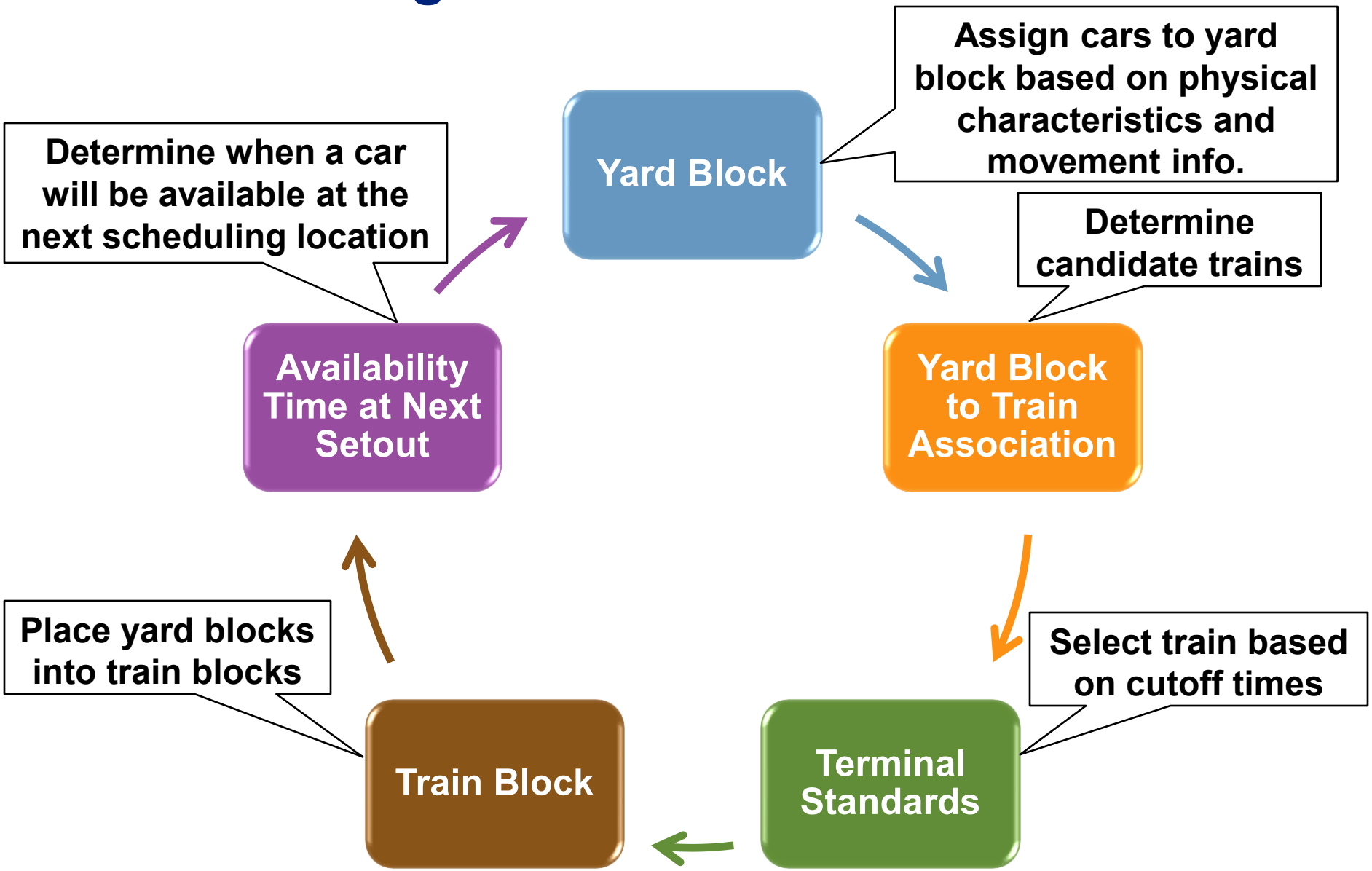
Car Scheduling

- As a railroad, Union Pacific generates revenue by moving railcars from one location to another.
- **Car Scheduling** is, therefore, one of the railroad's **most critical operations**.
- Car Scheduling can be defined as the process by which **trip plans** are generated for **individual cars**.
- Driving force for:
  - **Planning tools** for yard management
  - **Hump** yard and **switching** operations
  - **Consist** documentation
  - Crew **work orders**
  - **ETA/ETI** for customers and industry
  - **Compliance** with physical **track restrictions**

Trip Plans

- **Trip plans provide information about:**
  - **Route cars will traverse.**
  - **Estimated time cars will be at specific route locations, particularly ETA at destination.**
  - **Trains on which cars will move.**
  - **Pickup/Setout location and type (station/delivery/industry)**
  - **Hold locations, i.e. yards at which cars must be held and await manual intervention.**
  - **Cutoff times, the times by which cars must arrive to adhere to their schedule.**
  - **Order in which cars are placed on a train:**
    - **Area of the train (front/middle/rear)**
    - **Adjacent cars**

Car Scheduling Process



Yard Block Assignment

Yard Blocks

- **Yard blocks** group cars with similar characteristics together so they can move as **a single unit**.
- Critical to **efficient yard management**. Essential to **reducing car handlings**.
- **Route cars** along "**shortest**" **paths**.
- Ensure that cars take "**legal**" **routes**:
  - High/Wide Violations
  - Speed Restrictions
  - Heavy Cars
  - Long-Term Detours
- Satisfy individual **customer's needs/requests**.

Yard Block Types

- **Four types** of yard blocks:
  - **Schedule** – for cars in **normal movement** status.
  - **Industry** – for cars ready to be pulled **from industry**.
  - **Hold** – for cars in some type of **hold status**.
  - **Intermodal** – used only for **intermodal traffic**.

- **Non-scheduled blocks** are a sub-category of each block type that do not have associated trains.

Yard Block Definitions

- Define which cars get assigned to a yard block.
- Expressed using **include**/**exclude** "groups".
- Examine **car characteristics** (>30 attributes):
  - **Shipment** characteristics:
    - Origin/Destination
    - Load/Empty Status
    - Commodity
    - Shipper/Consignee
    - Current Location
    - Inbound Train
  - **Physical** car characteristics:
    - Initial/Number
    - Length
    - Car Kind
    - Speed Restrictions
  - **Miscellaneous** characteristics:
    - Special Condition Codes
    - Car Owner
    - Car Pool
    - Movement Authority

Example: A “Simple” Yard Block Definition

```
** Block ID DEMS / Definition 3 **
 Effective Dates: 07/24/2007 00:00 – END
 SCHEDULE BLOCK

 * GROUP NUMBER 1 – Inclusion
   Destination BKP:    From 013 To 023
                       From 081 To 081
   System Destination:    From JF629 To JF637
                          From MB000 To MB036
                          From MC002 To MC039
                          From MX018 To MX067
                          From X 012 To X 081
                          From XA058 To XA058
   Destination Circ7/ZTS:  MX001  From 0100000 To 0499999
   Comments:  ALS TRAFFIC
```

Example: A “Complex” Yard Block Definition

```
** Block ID DEMS / Definition 9 **
 Effective Dates: 04/29/2010 09:00 - END
 SCHEDULE BLOCK

* Group Number 1 - Exclusion *
  System Destination:  BF005
                       SO143
* Group Number 2 - Exclusion *
  Speed Restriction:  R40
* Group Number 3 - Exclusion *
  Destination BKP:  072
  Car Kind:  C4
  Comments:  NEED TO MOVE SAME AS LOADED C5T
* Group Number 4 - Inclusion *
  System Destination:   From A 165 To A 223
                        From AD004 To AD004
                        From BV001 To BV011
  Comments:  SPRING TRAFFIC (FORMERLY NEFF BLOCK)
* Group Number 5 - Inclusion *
  System Destination:   From MK566 To MK567
                        From MK573 To MK573
  Destination Circ7/ZTS:  MK641  From 0380400 To 0380499
                          MK641  From 0381600 To 0381699
                          MK594  From 0541600 To 0541699
                          MK583  From 0574000 To 0574099
                          MK610  From 0574900 To 0574999
  Comments:  MCALESTER TRAFFIC
```

```
* Group Number 6 - Inclusion *
  Destination BKP:   From 072 To 075
  System Destination:   From A 224 To A 280
                        From BA001 To BA001
                        From BV032 To BV059
                        From FL105 To FL186
                        From GB020 To GB020
                        From GV004 To GV044
                        From HB002 To HB003
                        From HB007 To HB013
                        From HI010 To HI010
                        From HP010 To HP016
                        From HR004 To HR058
                        From JH845 To JH850
                        From LF372 To LF372
                        From LS372 To LS373
                        From LT102 To LT110
                        From LT705 To LT705
                        From SL372 To SL372
                        From SO000 To SO109
                        From SO086 To SO143
  Destination Circ7/ZTS:  B 372  From 0300000 To 0599999
                          B 372  From 1100000 To 1399999
                          B 372  From 1500000 To 1699999
                          B 372  From 6100000 To 6299999
  Interchange Delivery:   JCT B 372 BOTH PTRA ORC 01
                          JCT HP014 BOTH PTRA ORC 01
                          JCT B 372 BOTH PTRA ORC 02
                          JCT HP014 BOTH PTRA ORC 02
                          JCT B 372 BOTH PTRA ORC 03
                          JCT HP014 BOTH PTRA ORC 03
                          JCT B 372 BOTH PTRA ORC 06
                          JCT HP014 BOTH PTRA ORC 06
                          JCT B 372 BOTH BNSF
                          JCT GV022 BOTH BNSF
  Comments:  BKP 71 IS SPLIT FOR HOUSTON IRN HOLD CARS MUST
              DRIVE TO B 372
```

Yard Block Assignment

- Every **scheduling location** has multiple yard block definitions that assign cars to particular yard blocks.
- Yard blocks definitions are first **filtered** based on their type and the status of the car being scheduled.
- Yard blocks definitions are ordered by **priority**: the highest-priority definition that applies is assigned.
- Each yard block may have **multiple definitions** that assign cars under different circumstances.
- Yard blocks **route cars** in particular directions by (indirectly) dictating the **train** they will run on and where their **next setout location** will be.

Example: Yard Blocks in the Council Bluffs Yard

BlkID	Def#	Type	Scheduled	
BOSP	1	Hold Block	NonScheduled	
BOEX	1	Hold Block	NonScheduled	
BOSL	1	Hold Block	NonScheduled	
BOHV	1	Hold Block	NonScheduled	
BOBO	1	Hold Block	NonScheduled	
UPFE	1	Hold Block	NonScheduled	
UPFE	2	Hold Block	NonScheduled	
EMCH	1	Hold Block	NonScheduled	
EDIH	1	Hold Block	Scheduled	
WODY	2	Hold Block	Scheduled	
OMAT	2	Hold Block	Scheduled	
CBNY	3	Hold Block	Scheduled	
ICTF	1	Intermodal Block	Scheduled	
NPSW	2	Intermodal Block	Scheduled	
MRTN	3		Scheduled	
FTWR	2		Scheduled	
DEMS	2		Scheduled	
IAIS	1		Scheduled	
MRTN	4		Scheduled	01/01/1997 09/01/2011
CN	1		Scheduled	
BRAN	1		Scheduled	
BNSF	1		Scheduled	
SHOP	1		Scheduled	
UGRE	1		NonScheduled	
UGRN	1		NonScheduled	
WODY	1		Scheduled	
DYBD	1		Scheduled	
INDU	1		Scheduled	
BARS	1		Scheduled	
YB75	1		Scheduled	
YB72	1		Scheduled	
DIXI	1		Scheduled	
CHRY	1		Scheduled	
MILL	1		Scheduled	
Y62M	1		Scheduled	
NATI	1		Scheduled	
DARL	1		Scheduled	
YB62	1		Scheduled	
CMTL	1		Scheduled	
RMP1	1		Scheduled	
OMAT	1		Scheduled	
TERR	1		Scheduled	
LV41	1		Scheduled	
MOVY	1		Scheduled	
VALY	1		Scheduled	
ICTF	2		Scheduled	
NPSW	1		Scheduled	
LN43	1		Scheduled	
CBCO	1		Scheduled	
NFLK	1		Scheduled	
CLBS	1		Scheduled	
CLBS	3		Scheduled	
NPLT	1		Scheduled	
ATCH	1		Scheduled	
DEMS	1		Scheduled	
FTWR	1		Scheduled	
FTWR	4		Scheduled	
DEMS	9		Scheduled	
FTWR	3		Scheduled	
DEMS	6		Scheduled	
DEMS	7		Scheduled	
DEMS	4		Scheduled	
MRTN	1		Scheduled	
CBPR	1		Scheduled	
EGLE	1		Scheduled	
BVLY	1		Scheduled	
CLTN	1		Scheduled	
DEMS	5		Scheduled	
CBNY	1		Scheduled	
MRTN	2		Scheduled	
PRVS	2		Scheduled	
DEMS	3		Scheduled	
PRVS	1		Scheduled	
CBNY	2		Scheduled	
UGRN	4	Industry Block	NonScheduled	
BARS	2	Industry Block	Scheduled	
YB75	2	Industry Block	Scheduled	
YB72	2	Industry Block	Scheduled	
FFOM	1	Industry Block	Scheduled	
TIES	1	Industry Block	Scheduled	
YB86	1	Industry Block	Scheduled	
UGRE	2	Industry Block	NonScheduled	
UGRN	2	Industry Block	NonScheduled	
FOMP	1	Industry Block	Scheduled	
TIEP	1	Industry Block	Scheduled	
Y86P	1	Industry Block	Scheduled	
PUL1	1	Industry Block	Scheduled	
BARP	1	Industry Block	Scheduled	
Y75P	1	Industry Block	Scheduled	
Y72P	1	Industry Block	Scheduled	

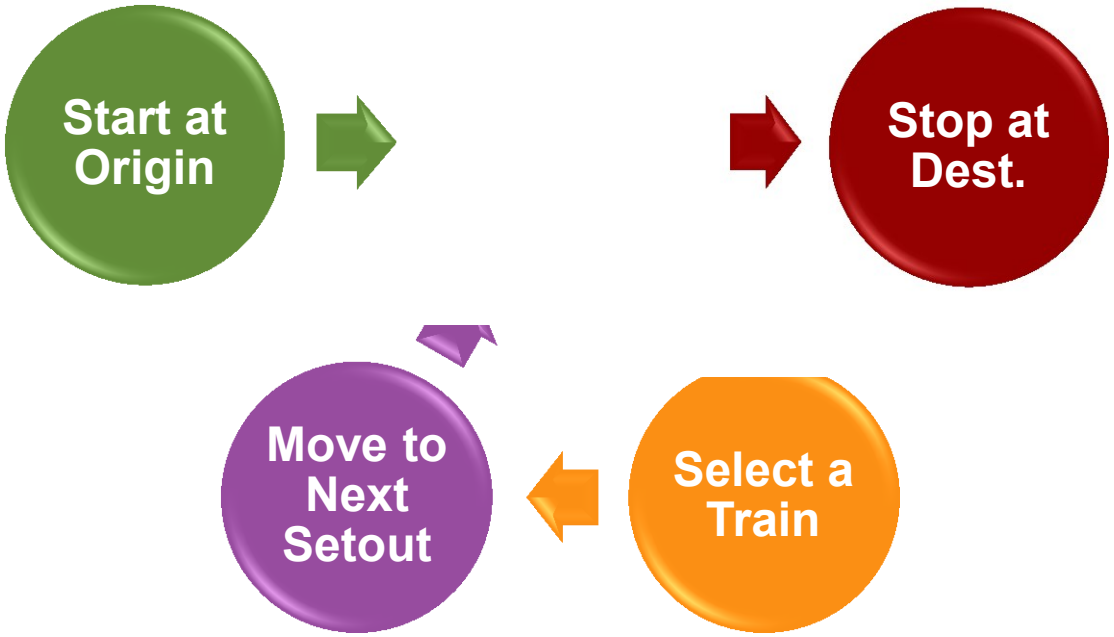
90 Yard Block Definitions

62 Distinct Yard Blocks

NetControl
Car Scheduling

TCS Car Scheduling

- **TCS** Car Scheduling is **deterministic**:
  - The **yard block assigned** is **always the same**.
  - The **next setout location** is **always the same**.
  - The **trip plan** of a car/waybill is **always the same**.
- **Procedural process**:



NetControl Car Scheduling

- **Optimized routing** that takes advantage of **transient opportunities**:
  - Evaluate the **state of the railroad** in real time.
  - **Cost-based routing** → different routes at different times.
  - Costs include **miles**, **car handlings**, **time**, and **capacity**.
- Implications:
  - **Multiple possible trip plans** are generated.
    The routing engine selects the "best" one.
  - **Multiple feasible yard blocks** at each location.
  - **Track restrictions** are evaluated directly against the cars.
  - Cars are **scheduled as a group**, not individually.

Implementation

Rules-Based Implementation

- **Examine ALL yard block definitions** in order to provide all ***feasible* yard blocks**, rather than just the highest-priority, applicable one.
- Checking each definition **increases the work** that must be performed tremendously.
- A **Rete-based pattern matcher** reduces this work by **reusing results** of identical comparisons across multiple yard block definitions.
- **SOA Service** that uses **embedded instances** of the **Jess** rules engine utility.

Service-Based Implementation

- **XMF-based service (XML/SOAP/JMS).**
- **Request:**
  - **All required physical characteristics about a car.**
  - **All relevant waybill information.**
  - **List of scheduling parameters:**
    - **Scheduling location (BDF)**
    - **Current station location**
    - **Inbound train**
    - **Availability time**
    - **Hold Status**
- **Reply**
  - **List of all feasible yard blocks for each set of scheduling parameters.**

Interfaces

- The existing **Train Schedule Builder** (**TSB**) system continues to be the **rules management UI** through which yard block definitions are maintained.
- Yard block definitions are loaded directly from the **TSB database** and injected into the rules engine.

Achieving Scalability

Car Scheduling Statistics

- ~80,000 railcars owned by Union Pacific Railroad.
- Many more owned by other railroads and customers.
- ~250,000 railcars operated on UP lines every day.
- ~65,000 new car cycles generated every day.
- Cars are **rescheduled frequently**, for a variety of reasons.
  - ~65% cars rescheduled one or more times.
  - ~3.18 scheduling requests per car.
- ~200,000 car scheduling requests per day.
- ~2.3 car scheduling requests/second.

Yard Block Statistics

- ~3.5 scheduling locations are examined on average per scheduling request.
- ~700,000 (3.5 x 200,000) yard block assignments/day.
- ~8 (3.5 x 2.3) yard block assignments/second.

- 1,180 scheduling locations (BDF).
- ~20 yard block definitions per BDF.
- Big yard → more definitions (>100 at major yards).
- Large yards occur most frequently in car schedules.
- ~20,000 yard block definitions total.

Minimum Performance Requirements

- Assume a **120% load** on the system.
  - ~**240,000** car scheduling requests per day.
  - ~**2.8** car scheduling requests/second.
  - ~**840,000** (3.5 x 240,000) yard block assignments/day.
  - ~**9.8** (3.5 x 2.8) yard block assignments/second.
- These values assume an **even distribution** throughout the day, they do not consider **peak operating hours**.
- The optimized routing engine may **increase the number of yard block assignments** for each scheduling request, as different route alternatives are considered – TBD.

Vertical vs. Horizontal Scalability

- **Vertical scalability** is concerned with **per-instance runtime**: time/request.
  - Usually addressed with **faster hardware**.
- **Horizontal scalability** is concerned with **throughput**: # requests/time.
  - Usually addressed with **more hardware**.
  - **Concurrent** processing.
- Too often, **scalability issues** are resolved using **horizontal solutions**, when the real issue is an **inefficient** and **slow** software implementation.
  - Higher Hardware Costs
  - Complicated Architecture
  - Difficult to Maintain
  - Additional Software

Performance of a Rules Engine

- **Key Factors:**
  - **Hardware**
  - **Pattern Matching Algorithm: Rete**
  - **Rule Optimization**
  - **Implementation**
    - **Rule Encoding**
    - **Fact Representation**
  - **Concurrent Processing**

Performance Test Data

- **7,798** randomly selected **examples** (unique CSN).
- **~3 scheduling locations** per example.
- **552 BDF locations referenced** by examples.
- **14,292** yard block definitions in referenced locations.
- **19,477** yard block groups in referenced locations.
- **Test data** loaded from network drive via **JDBC**.
- **IRN**s and **DRN**s loaded from LocationMaster via **XMF service calls**.
- **Yard Blocks** and **Super Block Keypoints** loaded from development CSG database via **JDBC**.

Performance Test

- Process all 7,798 examples
- Stand-alone batch process.
- Single-threaded
- Invocation:
  - One example per invocation.
  - All associated scheduling locations for each example.
- Caching:
  - Unlimited size
  - No eviction
  - No expiration
- 1GB maximum JVM heap size.
- Intel Core 2 Duo/2.33GHz/1.95GB RAM

Shadow Fact Implementation

Single Rule Engine

- Treats yard blocks as **data** using **shadow facts**.
- Individual yard block **conditions** are represented as **separate facts**. Hence, depending on the complexity of a yard block definition, it may be represented using many facts (> 20/30).
- **Few Rules** (~**30**) – Only high-level business logic.
- **Many Facts**:
  - **Yard block definitions**: groups/conditions
  - **Input data**: car characteristics and scheduling parameters
  - **Intermediate data**: satisfied definition/group/condition
  - **Output data**: evaluated locations with feasible yard blocks

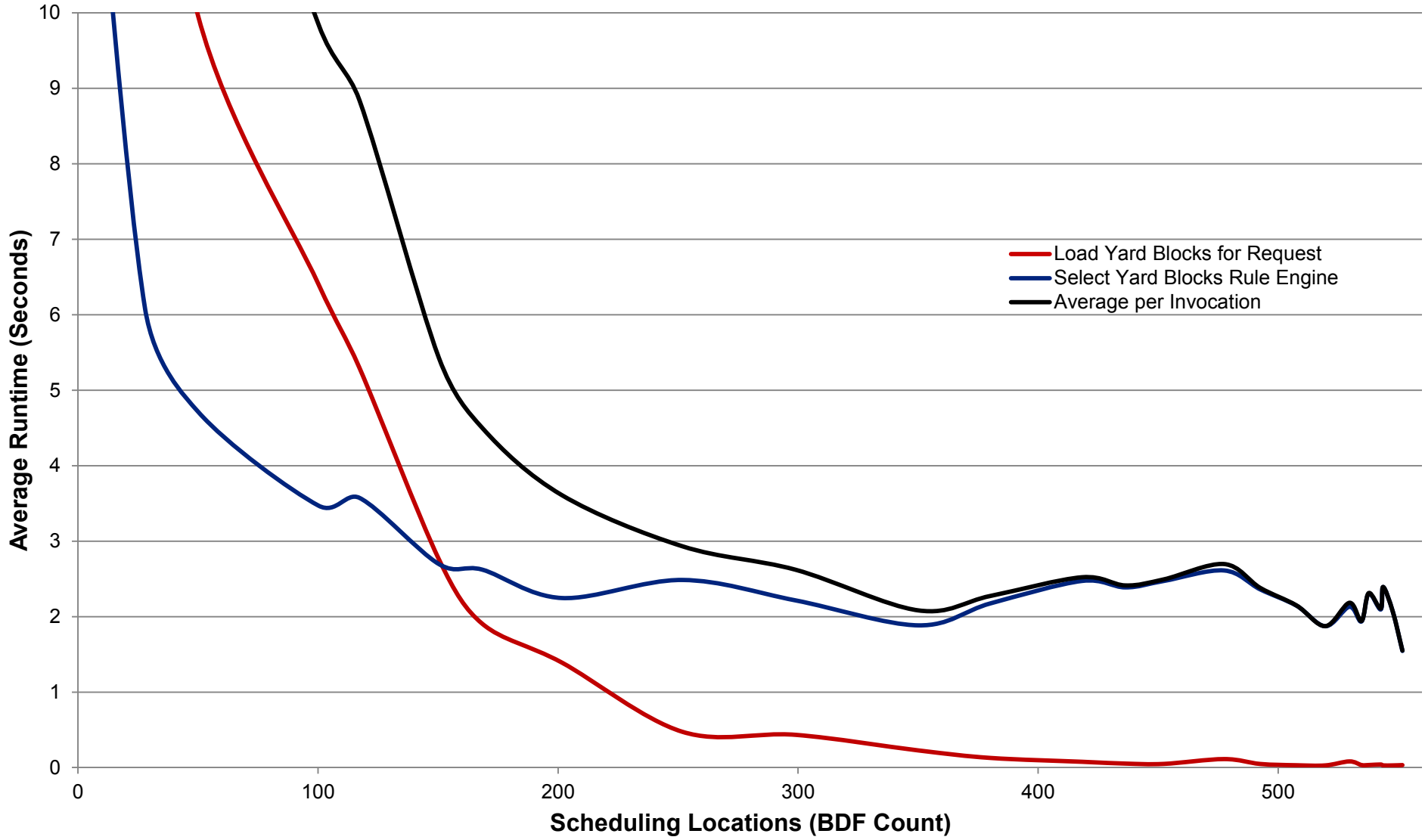
Shadow Fact Implementation

Single Rule Engine

- **Rule engine pool** – Each engine contains identical Jess code (rules, functions, templates, etc.), but a different set of yard block facts. **Peered** rule engines.
- **Two-level Yard Block Caching**:
  - **First cache** is on the **DAO** and preserves the yard block Java objects that are used as shadow facts.
  - **Second cache preserves partial matches** of yard block facts against rules in the rules engine. Yard blocks are added/removed from working memory based on this cache.
- **Selective Reset** – On reset, yard block facts stay in memory, preserving their matches to the rules, and all other facts (car, scheduling parameters, satisfied flags) are selectively retracted (slow).

Shadow Fact Implementation

Average Runtime per Invocation



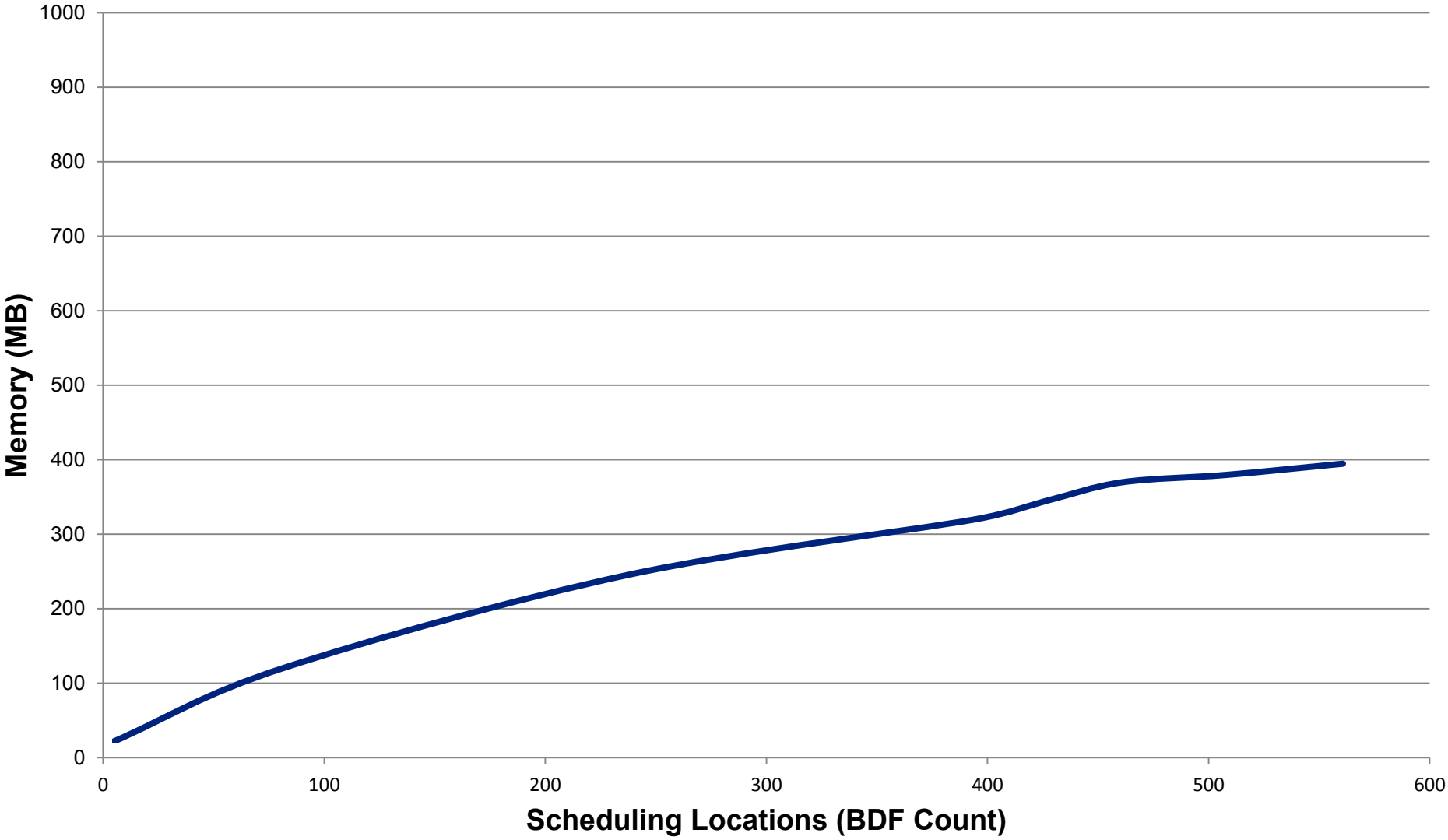
Shadow Fact Implementation

Average Runtime per Invocation

Task Name	Invoke Count	Total Elapsed	Lifetime Average	Last 1K Average	Last 1K Std. Dev.
Regression Test CFYB	7798	05:11:38.169	2.398s	2.008s	1.681s
Load Yard Blocks for Request	7798	00:14:36.488	0.112s	0.007s	0.087s
DAO - Get Yard Blocks	552	00:13:56.906	1.516s	1.516s	1.619s
Export Yard Blocks	552	00:00:09.479	0.017s	0.017s	0.029s
Select Yard Blocks Rule Engine	7798	04:57:01.570	2.285s	2.001s	1.681s
Rule Engine Input Data Export	7798	00:01:37.398	0.012s	0.010s	0.009s
Rule Engine Execute	7798	02:11:42.258	1.013s	0.883s	0.822s
Rule Engine Reset	7798	02:43:41.333	1.259s	1.108s	0.861s

Shadow Fact Implementation

Memory Consumption



Shadow Fact Implementation

Analysis

- **5.6 concurrent requests to achieve: 2.8 requests/second throughput at 2.001 seconds/request.**
- **Many partial matches are being created for scheduling locations other than those under consideration.**
- **Eliminating these partial matches may improve performance.**
- **Moving the yard block facts in and out of memory isn't an option since it will add a significant amount of work to re-match those facts against the rules each time.**
- **Solution? Specialized rule engines?**

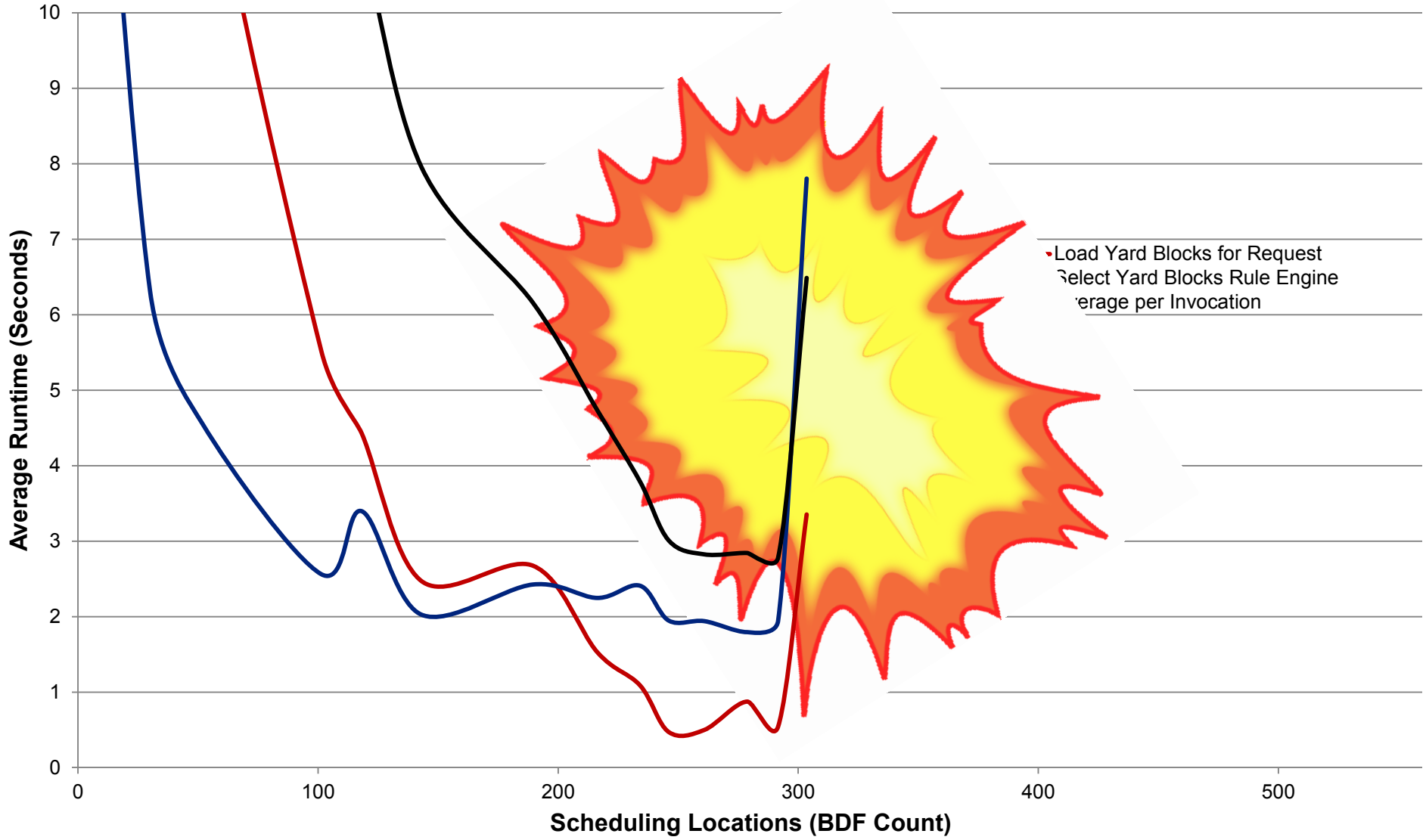
Shadow Fact with Specialized Rule Engines

Specialized Rule Engine Cache

- **Specialized rule engine cache**, keyed on the BDF.
- **Many Rule Engines** – Each scheduling location (BDF) has its own, dedicated, rule engine.
- **Few Rules** (~30) – Only high-level business logic.
- **Many Facts** – The facts are distributed across the many engines.
- **Multiple Rule Engine Invocations** – Each distinct BDF listed in the scheduling parameters requires a separate rule engine invocation.
- **Larger Memory Footprint** – ↑engines → ↑memory

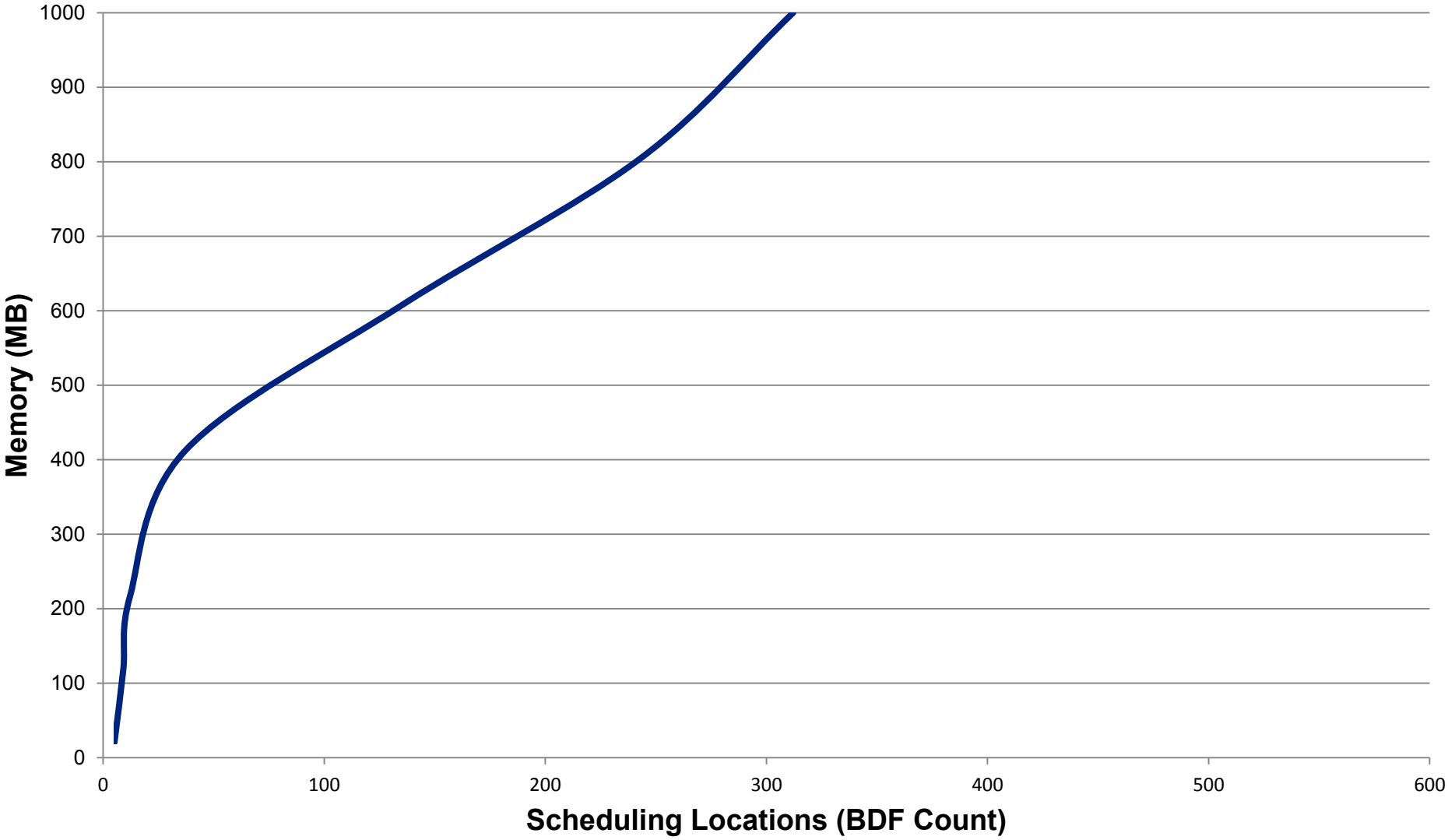
Shadow Fact with Specialized Rule Engines

Average Runtime per Invocation



Shadow Fact with Specialized Rule Engines

Memory Consumption



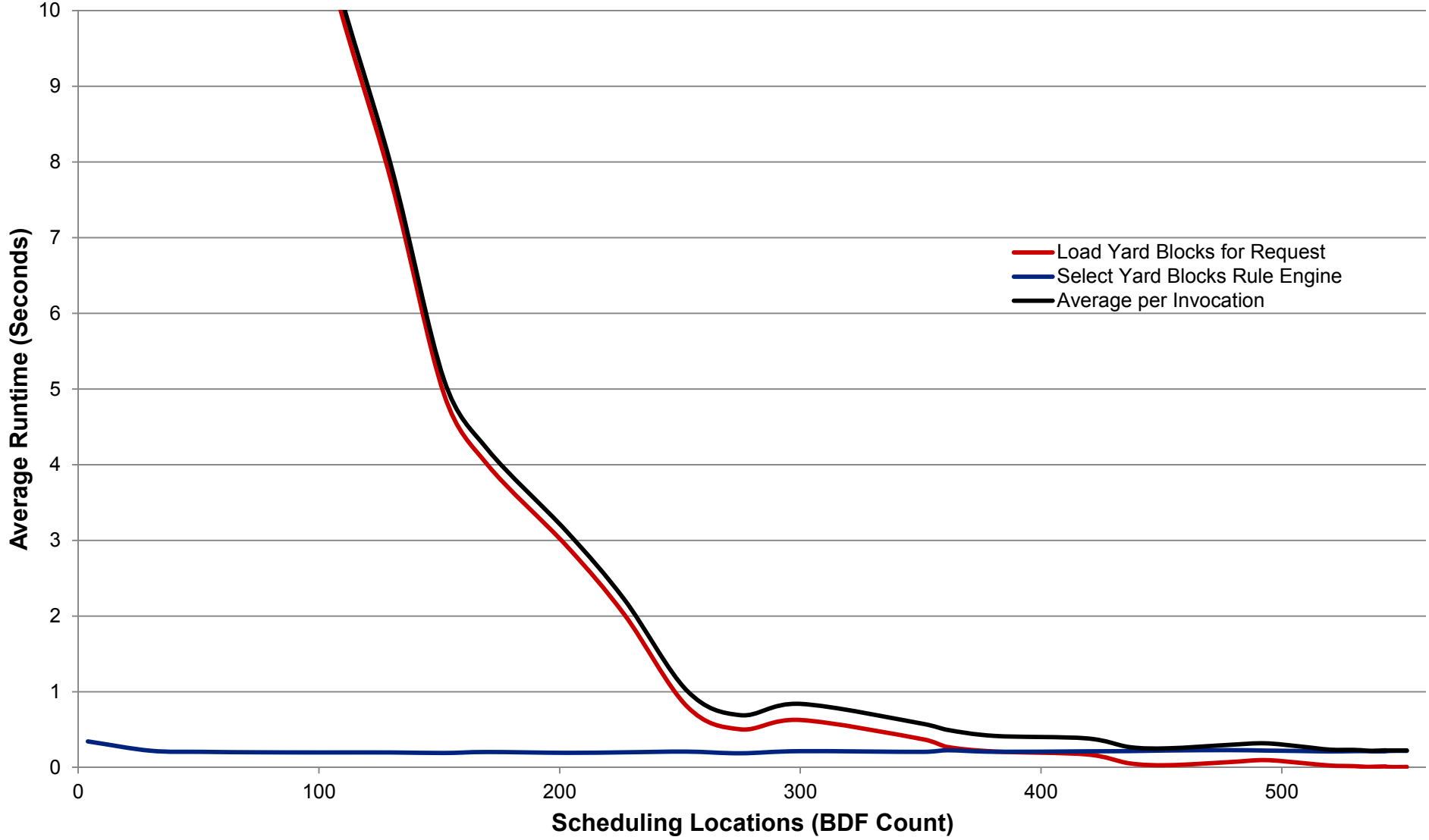
Source Generator Implementation

Single Rule Engine

- Treats yard blocks as **code**; transform to **Jess rules**.
- **Increases yard block load** time since it requires additional time to **generate the source script** (this processing can be done in an external process).
- **Many Rules** – one for each yard block definition.
- **Few Facts** – car, scheduling parameters, satisfied yard block definitions, and evaluated locations.
- **Smaller memory footprint** – similar conditions in the yard blocks are merged into single objects in the rule engine's pattern-matching (Rete) network.
- **Faster execution** of rule engine.
- **Slower export** of input data.

Source Generator Implementation

Average Runtime per Invocation



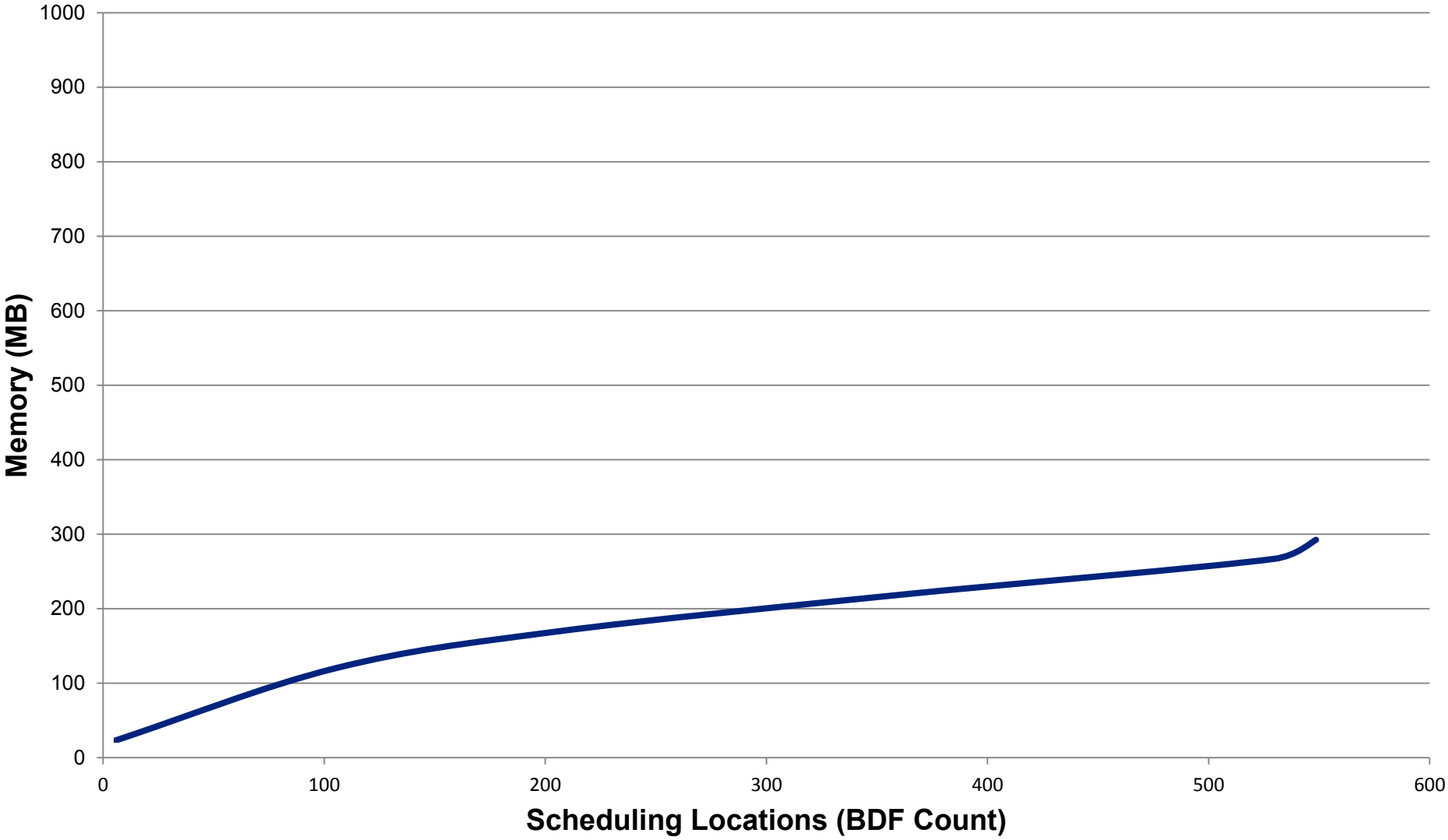
Source Generator Implementation

Average Runtime per Invocation

Task Name	Invoke Count	Total Elapsed	Lifetime Average	Last 1K Average	Last 1K Std. Dev.
Regression Test CFYB	7798	00:51:44.905	0.398s	0.224s	0.106s
Load Yard Blocks for Request	7798	00:23:46.225	0.183s	0.006s	0.085s
DAO - Get Yard Blocks	552	00:03:06.485	0.338s	0.338s	0.359s
Yard Block Jess Source Gen.	552	00:20:01.583	2.177s	2.177s	3.124s
Select Yard Blocks Rule Engine	7798	00:27:58.540	0.215s	0.218s	0.060s
Rule Engine Input Data Export	7798	00:15:29.731	0.119s	0.117s	0.057s
Rule Engine Execute	7798	00:03:31.774	0.027s	0.026s	0.015s
Rule Engine Reset	7798	00:08:56.816	00.069s	0.075s	0.009s

Source Generator Implementation

Memory Consumption



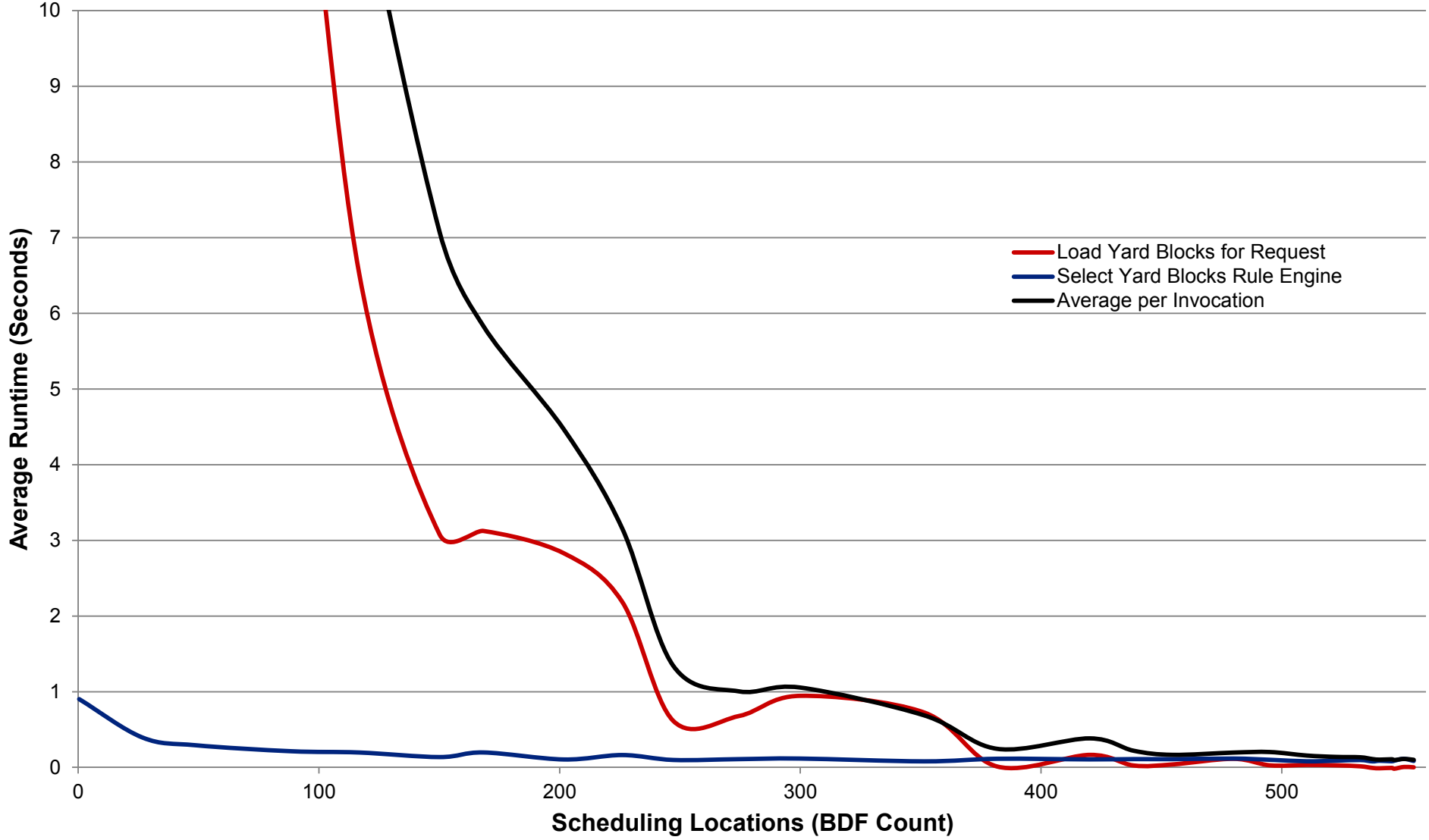
Source Generator with Specialized Rule Engines

Specialized Rule Engine Cache

- Yard blocks for each scheduling location are **transformed into Jess rules** and then loaded into a **specialized rule engine cache** keyed on the BDF.
- **Many Rule Engines** – Each scheduling location (BDF) has its own, dedicated, cached rule engine.
- **Moderate Number of Rules** – Each rule engine contains a rule for each yard block definition at the BDF location for which it is specialized.
- **Few Facts** – cars, scheduling requests, satisfied yard block definitions, and evaluated locations.
- **Multiple Rule Engine Invocations**
- **Larger Memory Footprint** – ↑engines → ↑memory

Source Generator with Specialized Rule Engines

Average Runtime per Invocation



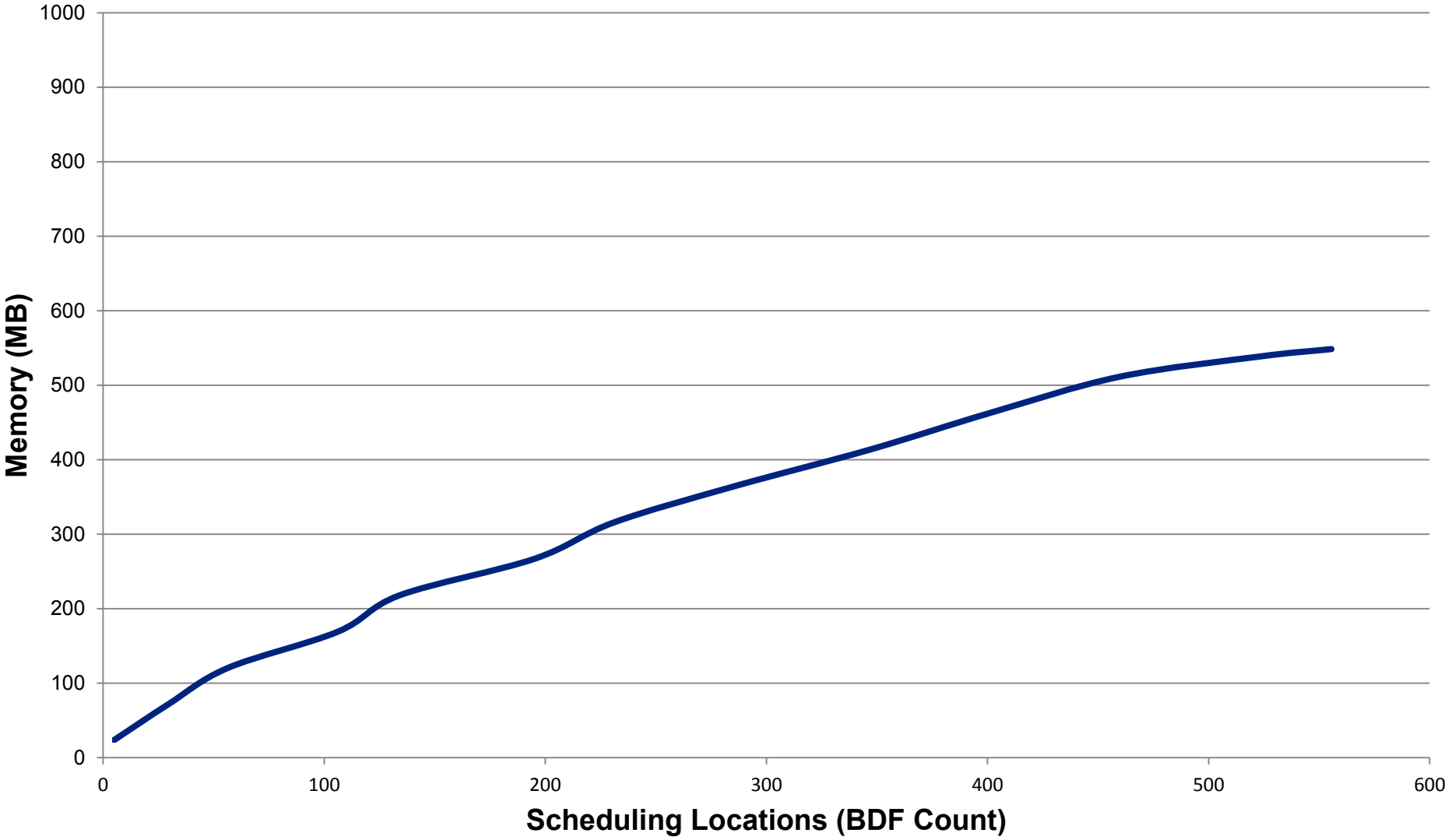
Source Generator with Specialized Rule Engines

Average Runtime per Invocation

Task Name	Invoke Count	Total Elapsed	Lifetime Average	Last 1K Average	Last 1K Std. Dev.
Regression Test CFYB	7798	01:00:31.523	0.466s	0.121s	0.197s
Load Yard Blocks for Request	7798	00:43:34.532	0.108s	0.007s	0.090s
DAO - Get Yard Blocks	552	00:16:17.770	1.771s	1.771s	2.105s
Yard Block Jess Source Gen.	552	00:21:18.609	2.316s	2.316s	3.574s
Export Yard Blocks	552	00:00:23.256	0.042s	0.042s	0.052s
Select Yard Blocks Rule Engine	24230	00:16:56.428	0.042s	0.041s	0.053s
Rule Engine Input Data Export	24230	00:16:41.160	0.041s	0.040s	0.053s
Rule Engine Execute	24230	00:00:06.730	0.000s	0.000s	0.002s
Rule Engine Reset	24230	00:00:07.927	0.000s	0.000s	0.002s

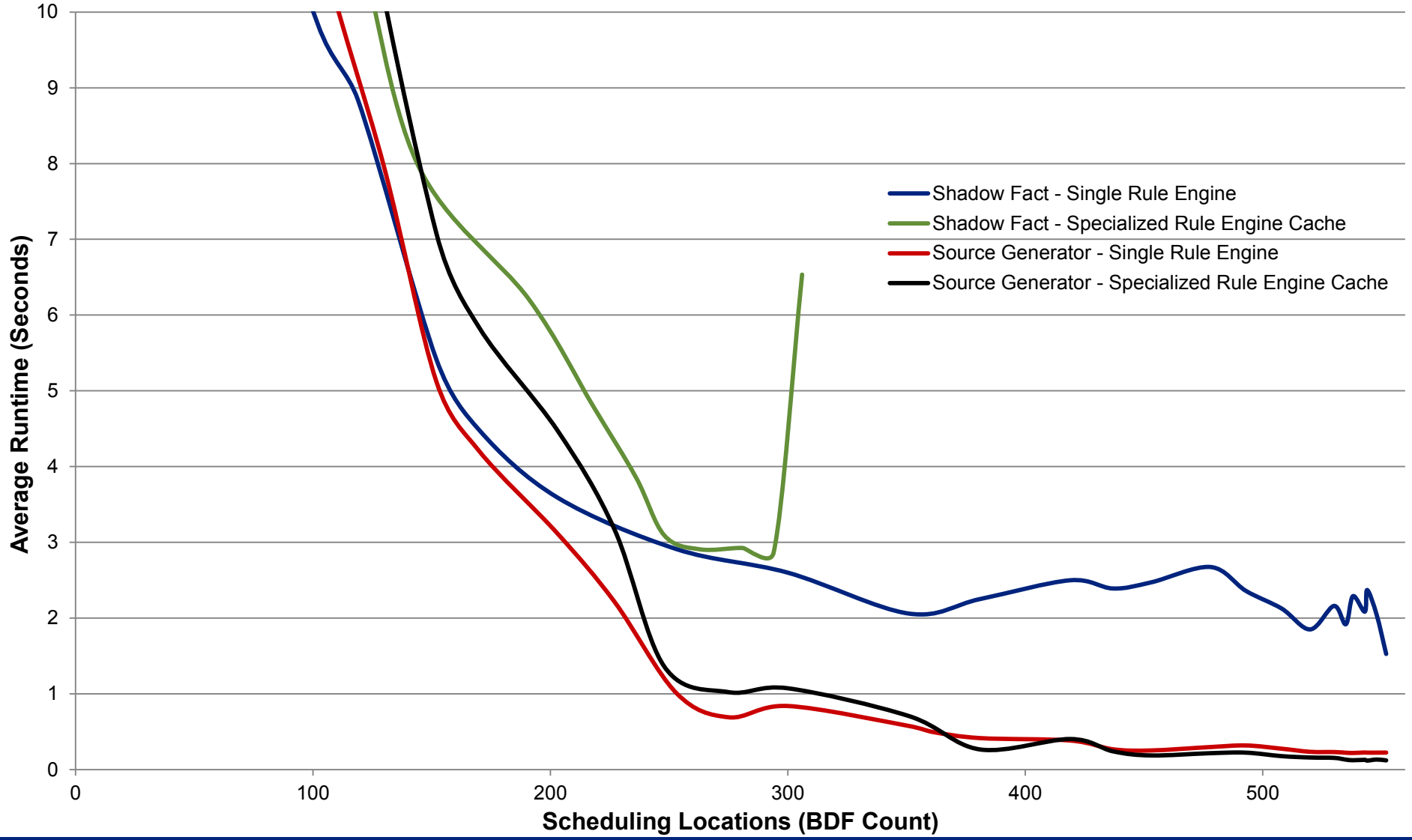
Source Generator with Specialized Rule Engines

Memory Consumption



Performance Summary

Average Runtime per Invocation Comparison



Performance Summary

- **Source generation** with **dedicated rule engines** is giving the **best runtime performance** with **reasonable memory consumption**.
- **Caching** of yard blocks only significantly improves performance after ~**300 BDF locations** are loaded.
- Using source generation in a single rule engine: the improved performance from yard block cache is offset by increase in runtime caused by large # of rules.
- **Externalizing source code generation** can improve performance at startup time.
- **Cache misses** typically load the less frequently used yard blocks: usually **small** and **fast** to load.
- A "**simple**", **service-based implementation** is sufficient to **satisfy scalability requirements**.

Thanks To:

- **IT Project Team**
  - **Manager: Michelle Gartner**
  - **Developers/Analysts**
    - **Ashley Wester**
    - **John Chekal**
    - **Mark Mordeson**
    - **Paul Thomas**
    - **Gordon Daugherty**
- **Network Planning**
  - **Mike Connolly, Pete Nipp**
  - **Service Design**
    - **Larry McClure**
    - **Tim Dorram**
    - **Candace Hughes**
    - **Bob Meder**
- **Decision Technologies**
  - **Michelle Clark, Shankara Kuppa**

Thank You