Enhancing Performance And Reliability of Rule Management Platforms

Mark Grechanik and B.M. Mainul Hossain
University of Illinois at Chicago

RulE Management Platforms (REMPs) allow software engineers to represent programming logic as conditional sentences that relate statements of facts (i.e., **rules**) using high-level declarative languages

Rule Structure And Example

Rules Format:

when
<*conditions*>
then
<*actions*>

Antecedent

Consequence

Example:

```
rule simple_rule:
  When
      Student( points >= 80)
  then
      System.out.println(“A+");
End
```

An Interesting Question

Rule A:	Rule B:
When Product (Price < 80) Then set(Price, 100);	When Product (Price > 50) Then set(Price, 200);

Price = 65

An Interesting Question

Rule A:	Rule B:
When	When
Product (Price < 80)	Product (Price > 50)
Then	Then
set(Price, 100);	set(Price, 200);

Price = 65

An Interesting Question

Rule A:	Rule B:

When
   Product (Price < 80)
Then
   set(Price, 100);
	When
	   Product (Price > 50)
	Then
	   set(Price, 200);

Price = 65

If-Then Decision Points

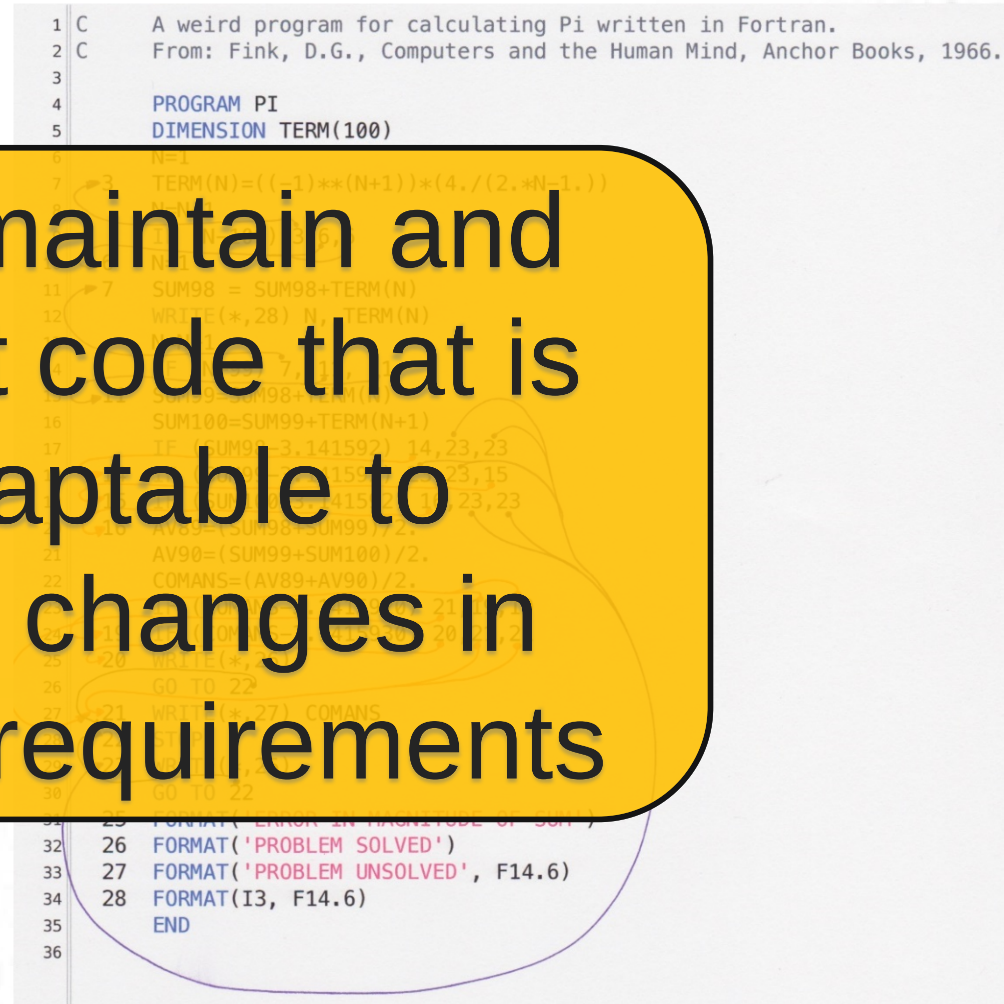
Alternative to rules is the imperative model where sequences of if-then statements with conditionals and loops are evaluated in a strictly defined order.

```
C     A weird program for calculating Pi written in Fortran.
C     From: Fink, D.G., Computers and the Human Mind, Anchor Books, 1966.

      PROGRAM PI
      DIMENSION TERM(100)
      N=1
    3 TERM(N)=((-1)**(N+1))*(4./(2.*N-1.))
      N=N+1
      IF (N-101) 3,6,6
    6 N=1
    7 SUM98 = SUM98+TERM(N)
      WRITE(*,28) N, TERM(N)
      N=N+1
      IF (N-99) 7, 11, 11
   11 SUM99=SUM98+TERM(N)
      SUM100=SUM99+TERM(N+1)
      IF (SUM98-3.141592) 14,23,23
   14 IF (SUM99-3.141592) 23,23,15
   15 IF (SUM100-3.141592) 16,23,23
   16 AV89=(SUM98+SUM99)/2.
      AV90=(SUM99+SUM100)/2.
      COMANS=(AV89+AV90)/2.
      IF (COMANS-3.1415920) 21,19,19
   19 IF (COMANS-3.1415930) 20,21,21
   20 WRITE(*,26)
      GO TO 22
   21 WRITE(*,27) COMANS
   22 STOP
   23 WRITE(*,25)
      GO TO 22
   25 FORMAT('ERROR IN MAGNITUDE OF SUM')
   26 FORMAT('PROBLEM SOLVED')
   27 FORMAT('PROBLEM UNSOLVED', F14.6)
   28 FORMAT(I3, F14.6)
      END
```

If-Then Decision Points

Alternative to rules is the imperative model where sequences of if-then statements with conditionals and loops are evaluated in a strictly defined order.



Hard to maintain and inefficient code that is not adaptable to frequent changes in business requirements

REMPS Are Widely Used

According to market report from Forrester, the estimated revenue of business rules management systems (BRMS) increased from $265 million in 2008 to over $600 million in 2011. It is one of the fastest growing markets.

One leading vendor, IBM ILOG Optimization is used by over 50% of the world's largest companies, 1000’s of Universities, and 1000's of application providers.

CLIPS

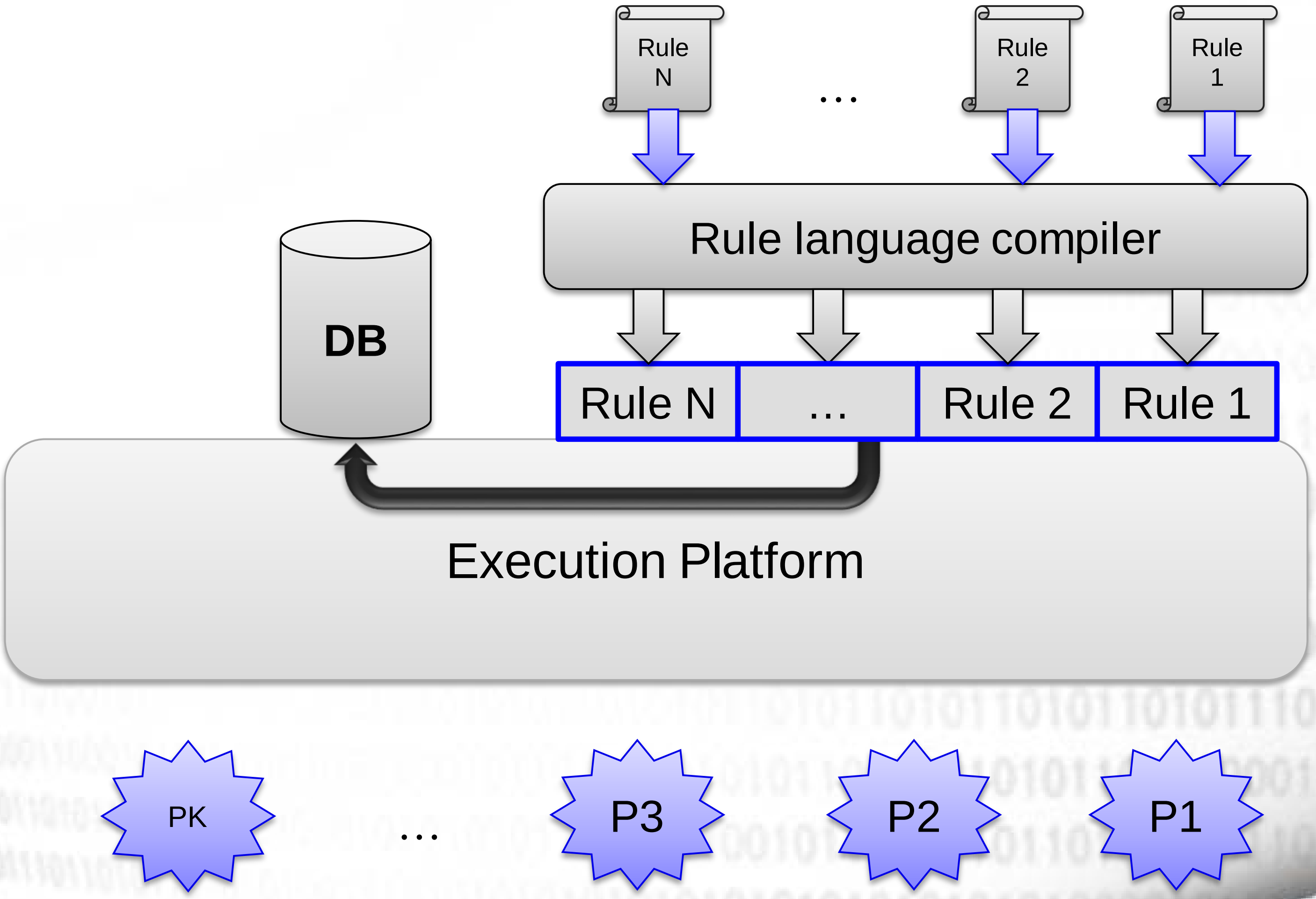
JBOSS DROOLS

BizTalk

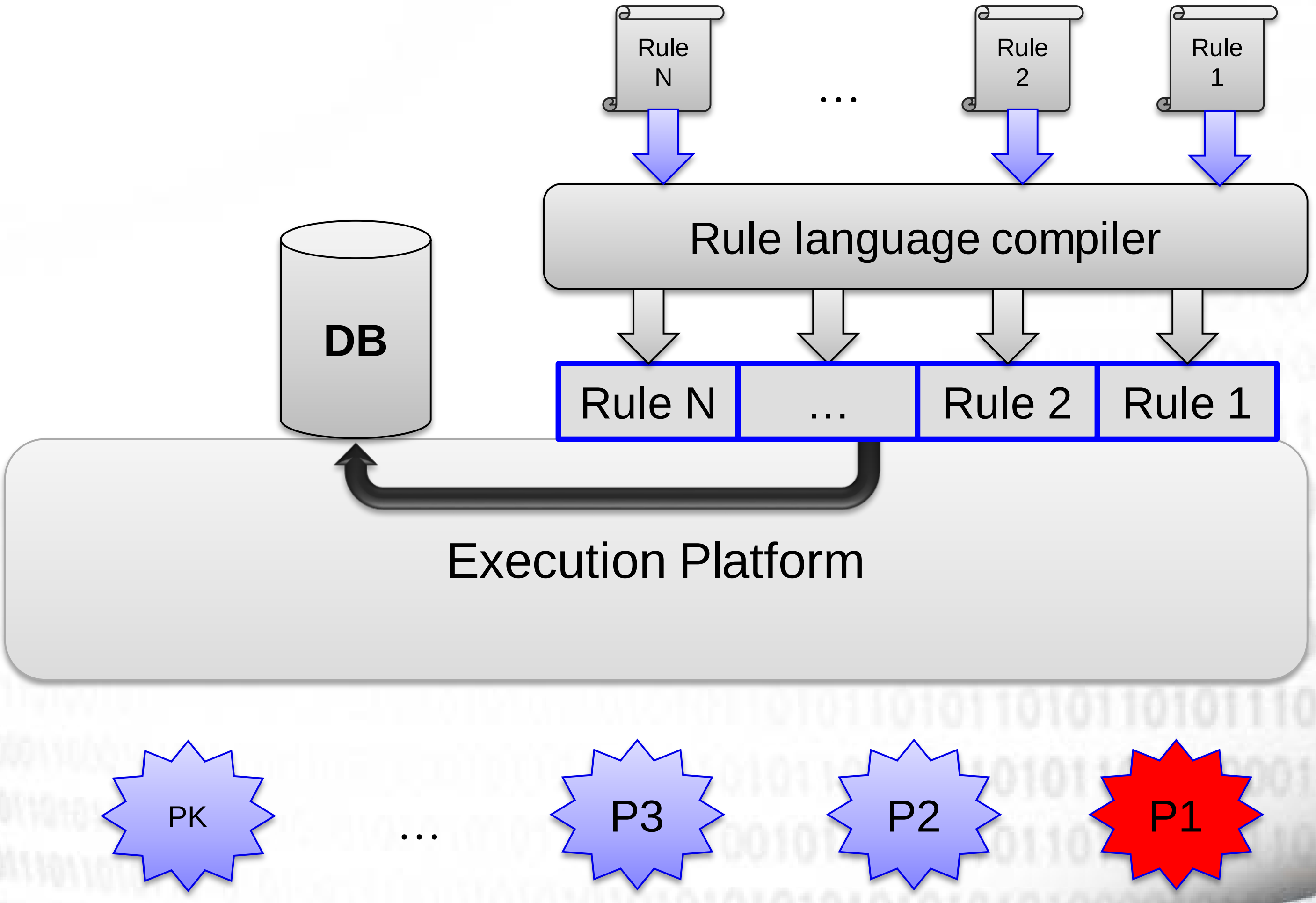
JESS

Oracle FUSION

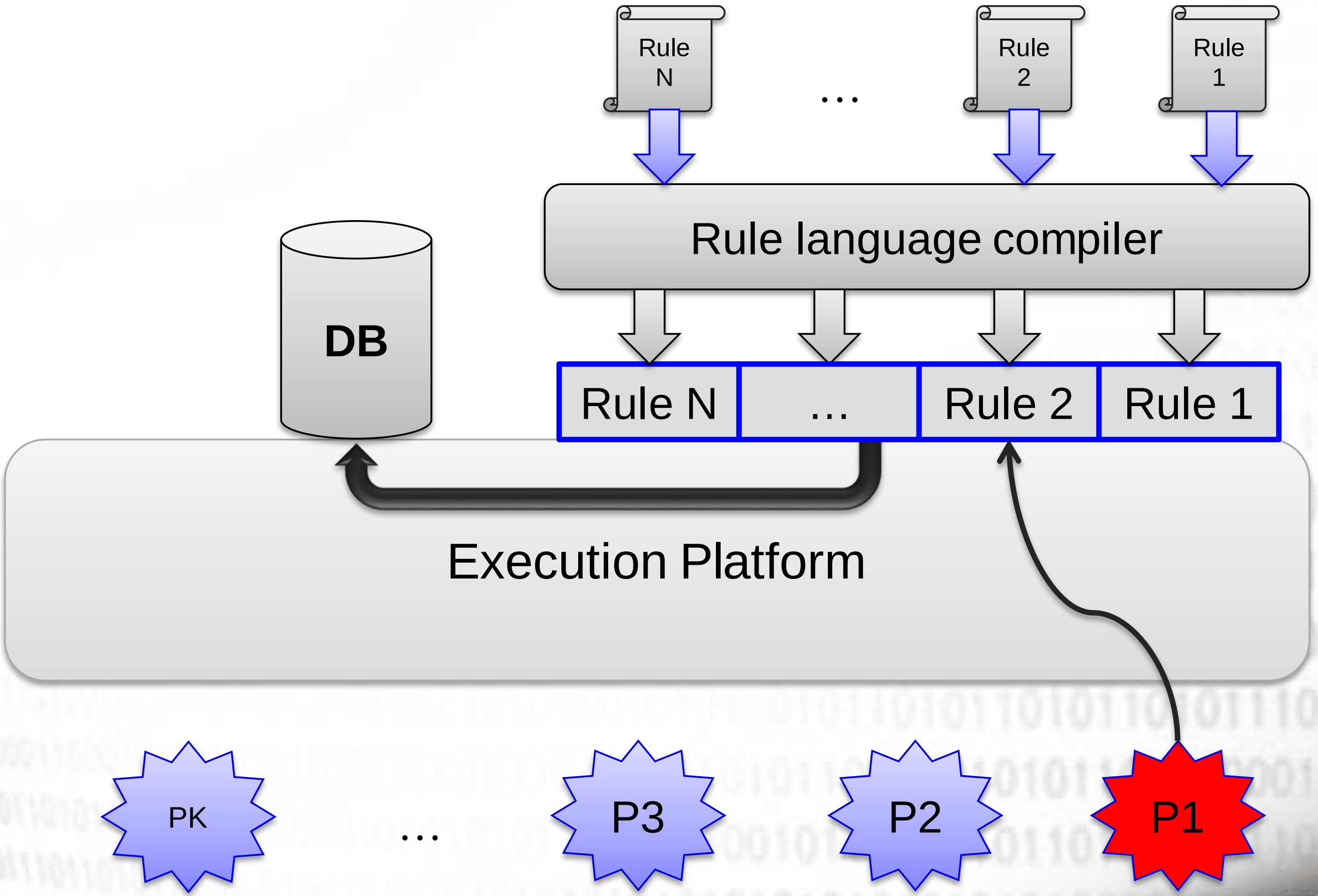
How REMPs Work



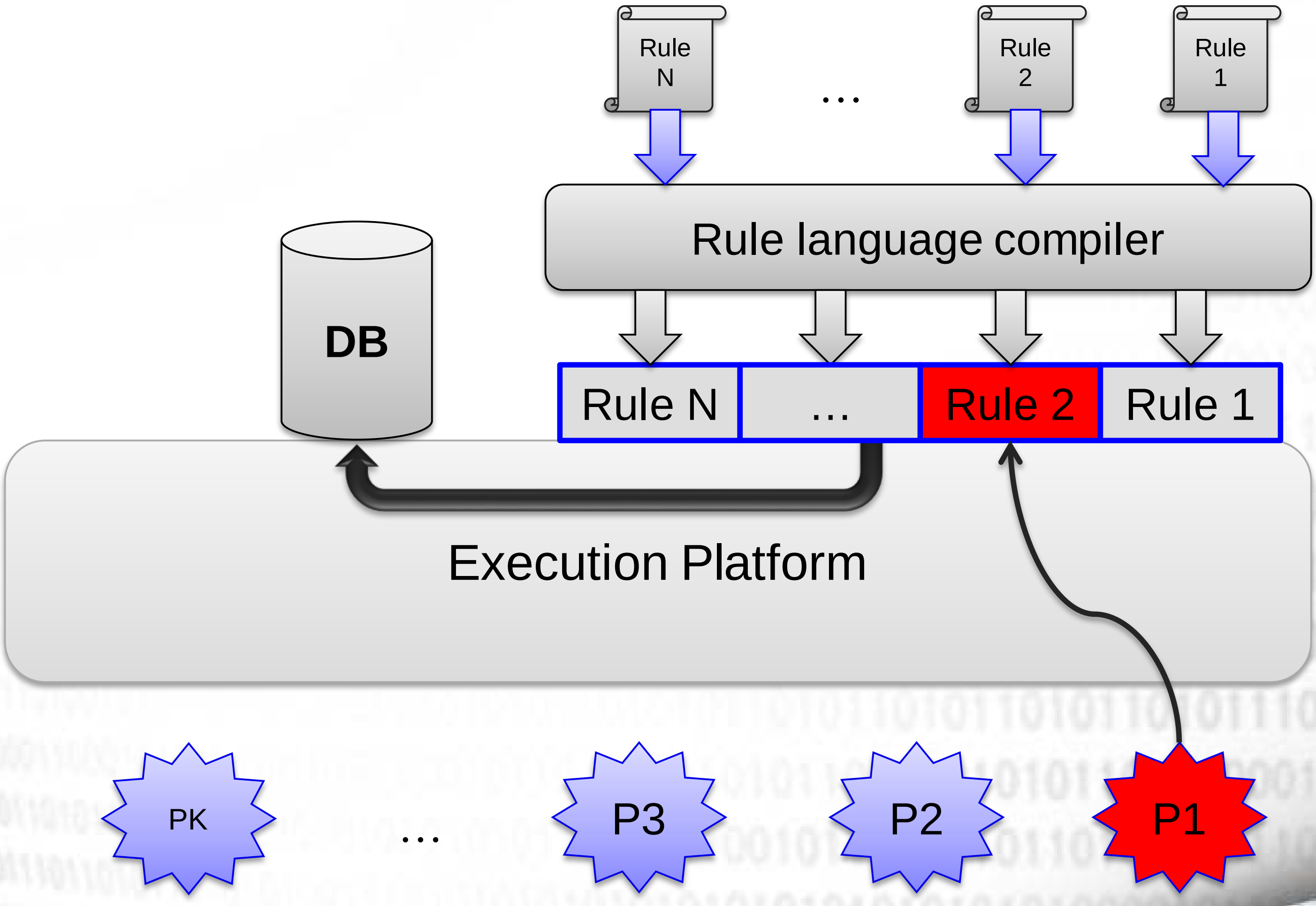
How REMPs Work



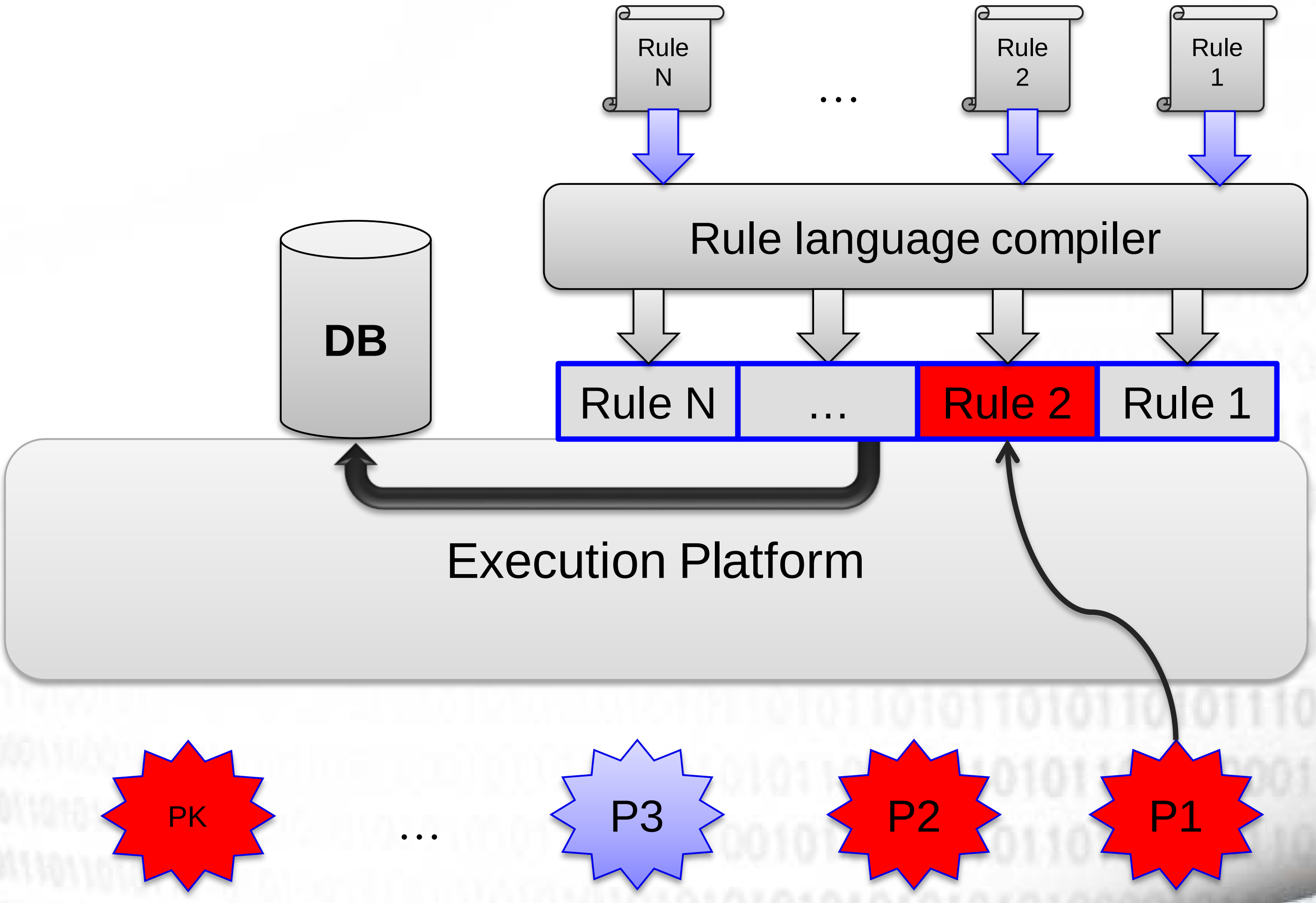
How REMPs Work



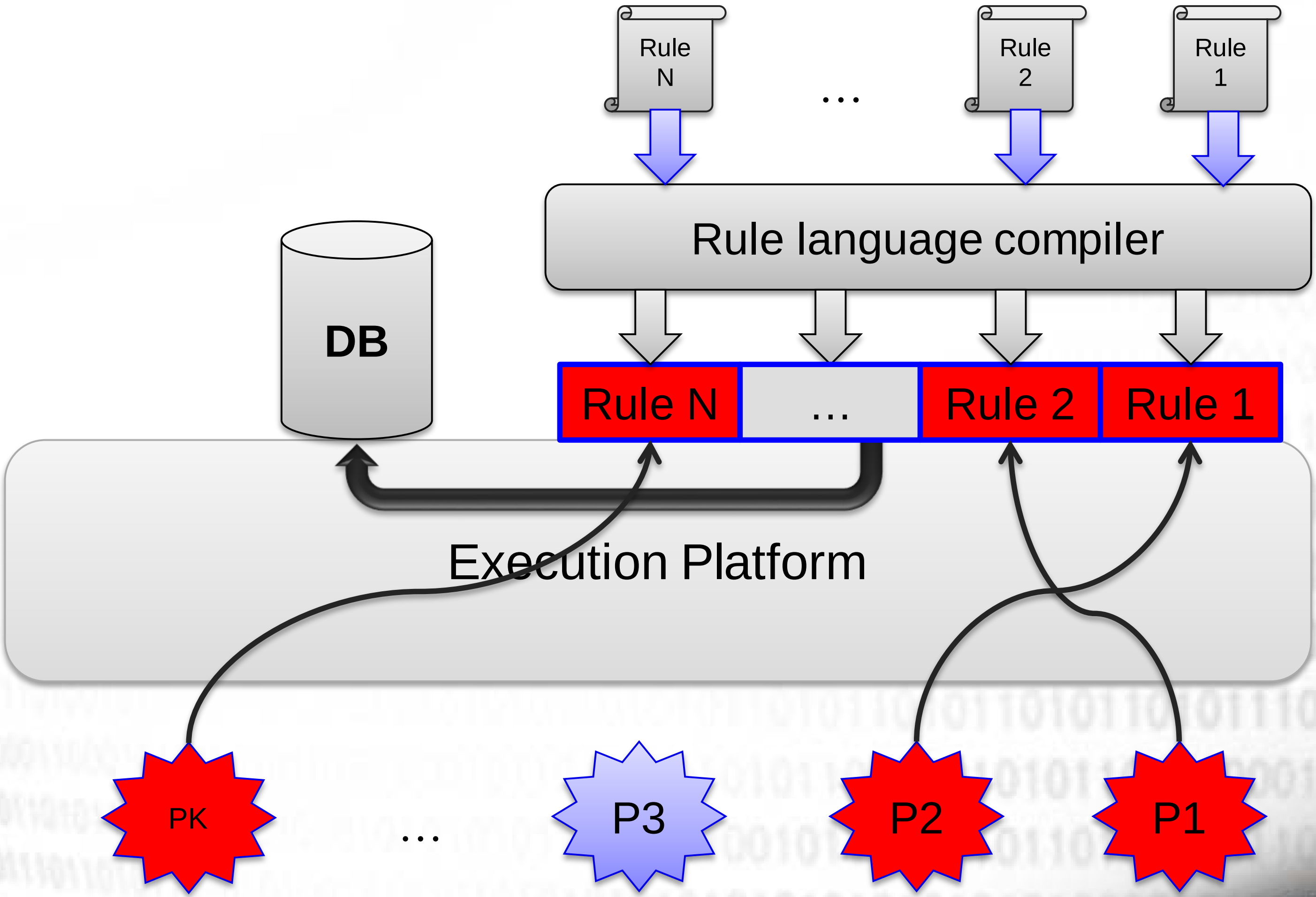
How REMPs Work



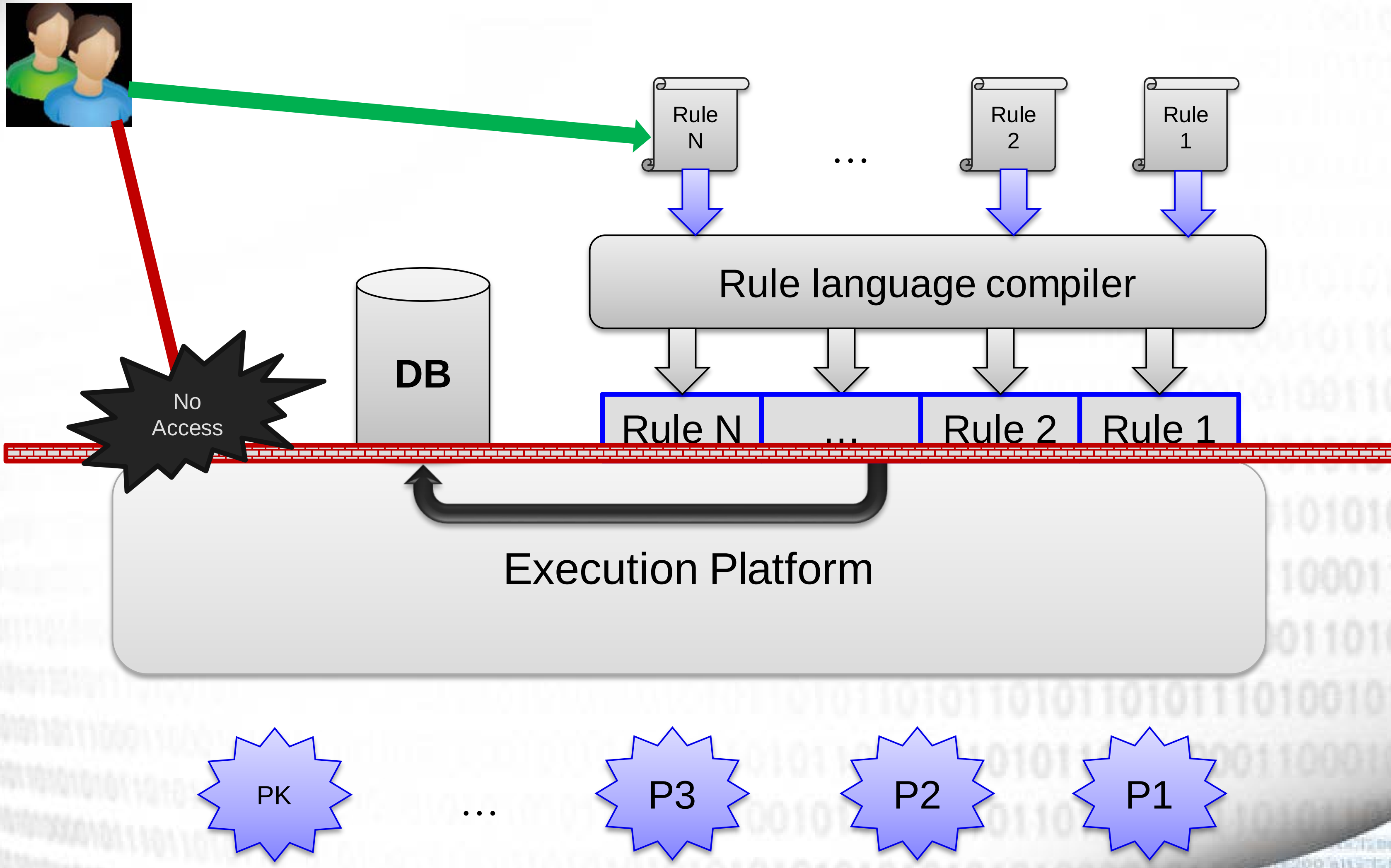
How REMPs Work



How REMPs Work



How REMPs Work



Separation of Concerns

A key property of REMPs is that they encapsulate the control flow that includes fact inference and rule firing logics

- It is a fundamental separation of concerns of the control flow and the rule business logic.

Software engineers concentrate on reasoning about higher-level business logic that they encode in rules without worrying about low-level details of rule invocations by effectively delegating this job to REMP engines

- Rule-driven APplications (RAPs) are highly adaptable to changing requirements, since stakeholders simply add new rules as independent modules to RAPs

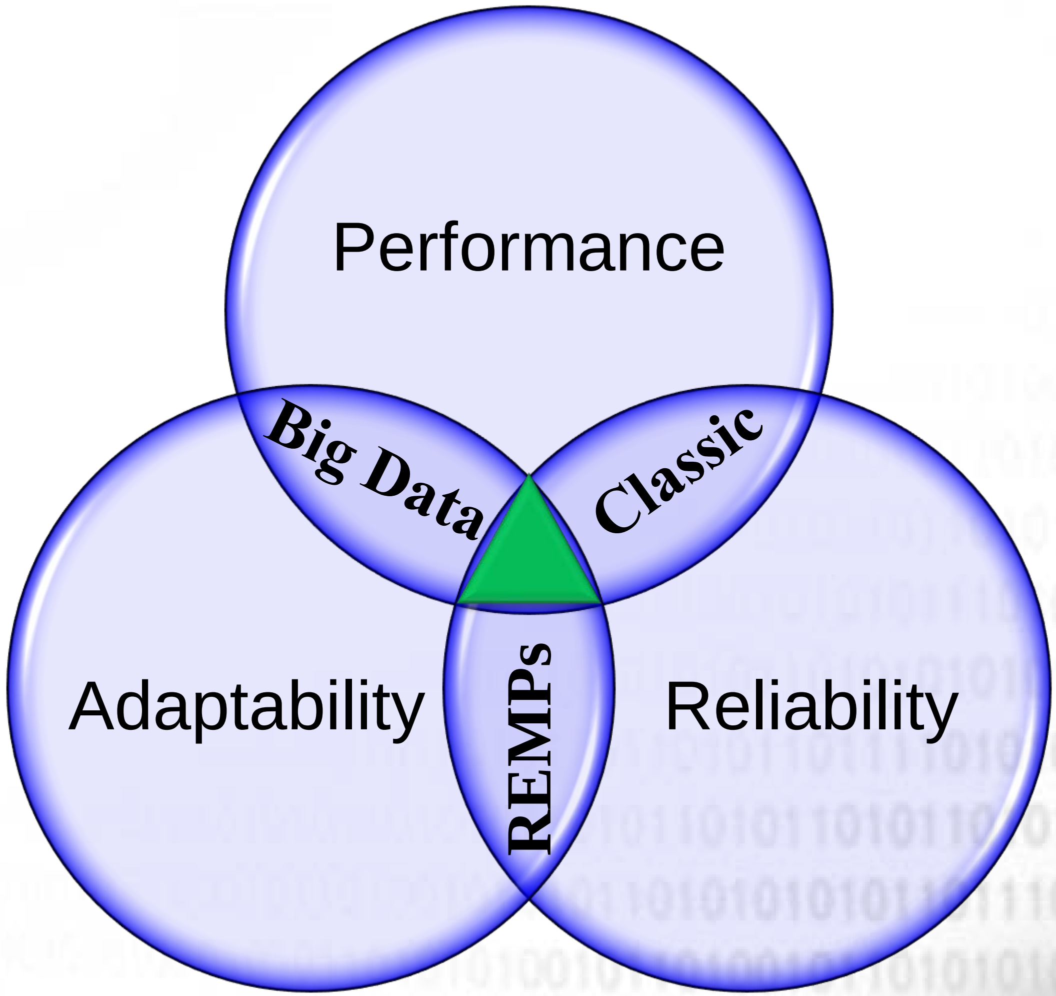
Benefits of REMPs And RAPs

- Easy to maintain & evolve
- Rules are easy to comprehend and highly modular
- RAPs are highly adaptable

Constraints of REMPs And RAPs

- RAPs may contain tens of thousands of rules
- Detection of conflicting rules is difficult
- Dependencies should not be introduced among rules

The PAR Model For REMPs



Fundamental Problem

- Many REMPs execute RAPs sequentially.
- Locks introduce complex dependencies among rules, thereby defeating the separation of concerns and eventually the adaptability of RAPs.
- How to enhance the performance of RAPs without sacrificing their adaptability and reliability?

Motivating Example: Credit

```
rule "Rule-Credit" salience 10
when
   $cashflow : Cashflow( $account:account,
   $date : date, $amount : amount,
   type==Cashflow.CREDIT )
   not Cashflow(account==$account,date<$date)
then
   //some code
   $account.setBalance(
      $account.getBalance()+$amount);
   retract($cashflow);
end
```

Motivating Example: Debit

```
rule "Rule-Debit" salience 1
when
    $cashflow : Cashflow( $account : account,
    $date : date, $amount : amount,
    type==Cashflow.DEBIT )
    not Cashflow(account==$account,date<$date)
then
    //some code
    if($account.getBalance()>$amount){
        $account.setBalance(
        $account.getBalance()-$amount); }
    else { new BlockedAccount($cashflow); }
    retract($cashflow);
end
```

Parallelism Interferes With Saliences

- Let us assume that a REMP engine executes rules in parallel and lock objects are used to synchronize concurrent accesses.
- Using a lock object effectively overrides the intention of the programmer to give the priority to the rule with a higher salience.
- Given the large number of possible interleavings among tens of thousands of rules in a RAP, it is very difficult to reason about interactions between saliences and synchronization lock mechanisms.

Reliability Meets Performance

In fully parallelized REMPs, loss or reliability comes from two sources:

- different orders in which rules are executed by the REMP engines;
- races between parallelized executions of rules.

When the system produces different results consecutively for the same computational task using the same input data, it is a serious problem, since it reduces the confidence of the user in the RAP and it impacts negatively the perception of the user about the business value that the company or organization delivers.

- the execution order for different instructions can be affected by multiple factors beyond the control of stakeholders
- sometimes even slight changes in the non-functional parameters of the environment (e.g., paging on demand) for executing RAPs result in different orders of instruction interleavings that lead to different results, hence the loss of reliability

The Problem Statement

- Enable REMPs to execute rules in RAPs in parallel
- Do not violate the separation of concerns in REMPs by requiring programmers to use synchronization lock mechanisms for concurrent accesses to shared resources
- Prevent races in parallelized RAPs without explicit using of locking mechanisms by programmers
- Choose a better schedule for executing rules that share the same lock objects to improve the overall performance of RAPs

Core Ideas

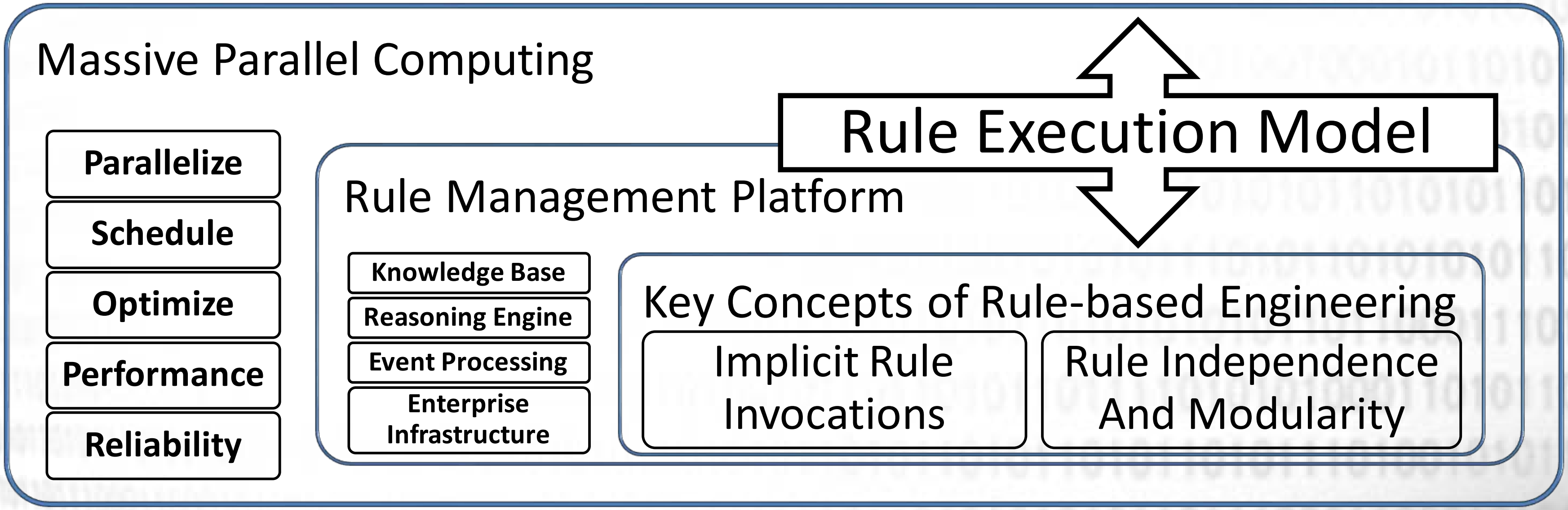
1) Find all concurrent access to resources from rules where one of the accesses is write.
2) Define synchronizations around these accesses.
3) Impose a complete ordering among all rules that are fired in working memory.

Our Solution - PERLATO

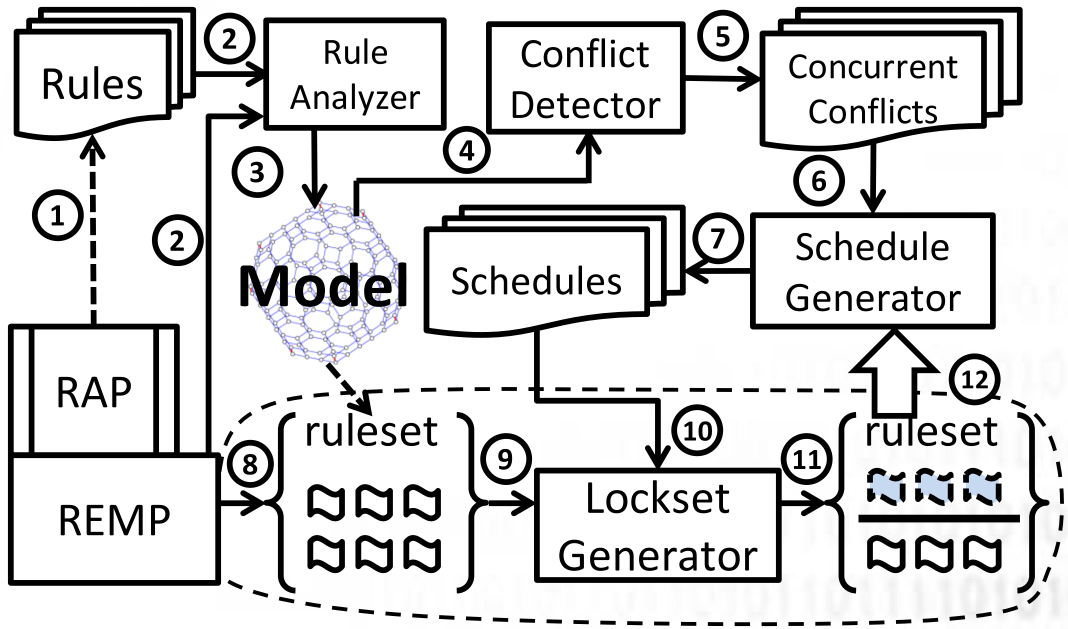
PErformance and Reliability for ruLe-driven ApplicaTiOns (PERLATO) connects separate layers or REMPs in a way that enable us to solve the fundamental problem of REMP.

- we obtain a rule execution model from a RAP that approximate different execution scenarios by using the if-then structure of rules by analyzing their antecedents and consequents
- the obtained rule execution model is used in PERLATO to detect races statically among these rules effectively and efficiently
- the rule execution model and locking strategies for a given RAP are passed to the REMP engine, so that it can precompute an execution schedule for rules in a RAP to optimize the performance of the RAP.
- **We implemented PERLATO for JBoss Drools, an open-source enterprise-level REMP and we evaluated PERLATO on three RAPs. The results suggest that PERLATO is effective and efficient, since we achieved up to 225% speedup on average without observing any races.**

Map fo PERLATO



The Architecture of PERLATO



Research Questions

Is PERLATO effective in achieving higher speedups for subject RAPs?

Is finer granularity locking strategy more effective in obtaining higher speedup for RAPs?

Is symbiotic scheduling effective in obtaining higher speedup for RAPs?

Result for Subject RAPs

Application	*Input Set*	*Facts*	*Rules Fired*	*Approach*	*Type*	*Avg*	*Med*	*Min*	*Max*	σ^2
EEWS	1	246896	1088	DROOLS	Seq	32.22	32.08	31.559	35.358	0.4
					Par	9.52	9.27	8.267	11.789	0.87
				PERLATO	Rule	13.44	13.6	10.927	14.911	0.99
					Atomic	10.85	10.82	10.21	11.867	0.14
					Variable	10.67	10.67	10.21	11.359	0.06
	2	334220	1454	DROOLS	Seq	42.31	42.12	41.508	43.322	0.21
					Par	11.62	11.53	10.299	14.126	0.85
				PERLATO	Rule	18.86	19.02	15.794	21.549	2.07
					Atomic	15.62	15.58	15.016	16.476	0.11
					Variable	15.08	15.02	14.24	16.689	0.21
	3	247113	572	DROOLS	Seq	19.51	19.44	18.906	21.115	0.24
					Par	6.05	6.19	3.947	8.242	1.32
				PERLATO	Rule	8.33	8.22	7.025	9.442	0.42
					Atomic	7.24	7.13	6.837	8.259	0.13
					Variable	6.79	6.76	6.662	7.165	0.02
TAXC	1	16	95	DROOLS	Seq	20.4	20.32	20.023	20.97	0.06
					Par	19.52	19.52	19.115	20.072	0.07
				PERLATO	Rule	19.71	19.69	19.232	20.657	0.13
					Atomic	19.62	19.61	19.144	20.227	0.08
					Variable	19.55	19.52	19.077	20.161	0.08
	2	22	162	DROOLS	Seq	42.48	42.48	41.553	43.63	0.32
					Par	34.27	34.6	31.866	36.019	1.13
				PERLATO	Rule	36.87	36.83	36.016	37.812	0.19
					Atomic	36.86	36.77	36.007	37.775	0.21
					Variable	36.8	36.74	35.972	37.698	0.23
	3	10	66	DROOLS	Seq	11.78	11.73	11.549	12.344	0.04
					Par	9.85	9.87	9.116	10.314	0.05
				PERLATO	Rule	9.93	9.9	9.641	10.431	0.03
					Atomic	9.91	9.92	9.145	10.342	0.04
					Variable	9.98	9.97	9.309	10.761	0.06

Conclusions

We created a novel solution for enhancing performance and reliability of rule-driven applications.

The results suggest that PERLATO is effective, since we achieved over 225% speedup on average.

Thank You

Email: drmark@uic.edu