Polski Internet Optyczny – PIONIER 2004
Maj 19-20, 2004 Poznań

Analiza wydajności technik komunikacji z usługą kontroli dostępu opartą na specyfikacji RAD

Paweł Słowikowski ps@agh.edu.pl
Marcin Jarząb mj@agh.edu.pl
Krzysztof Zieliński kz@ics.agh.edu.pl

Cyfronet, AGH - Kraków

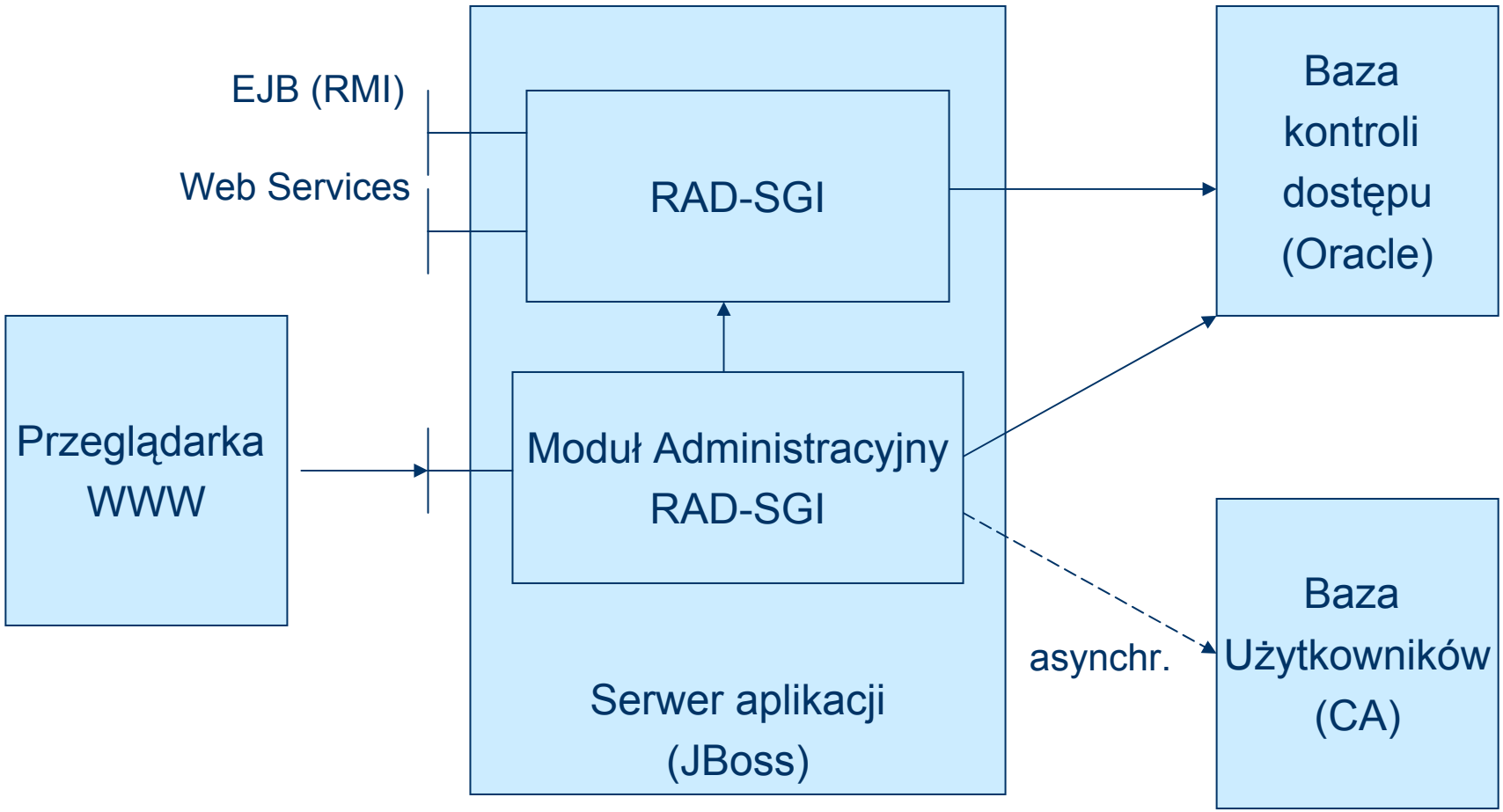
Plan prezentacji

- Architektura RAD
- Cel i opis testów
- Konfiguracja środowiska
- Tuning
- Wyniki
- Konkluzje

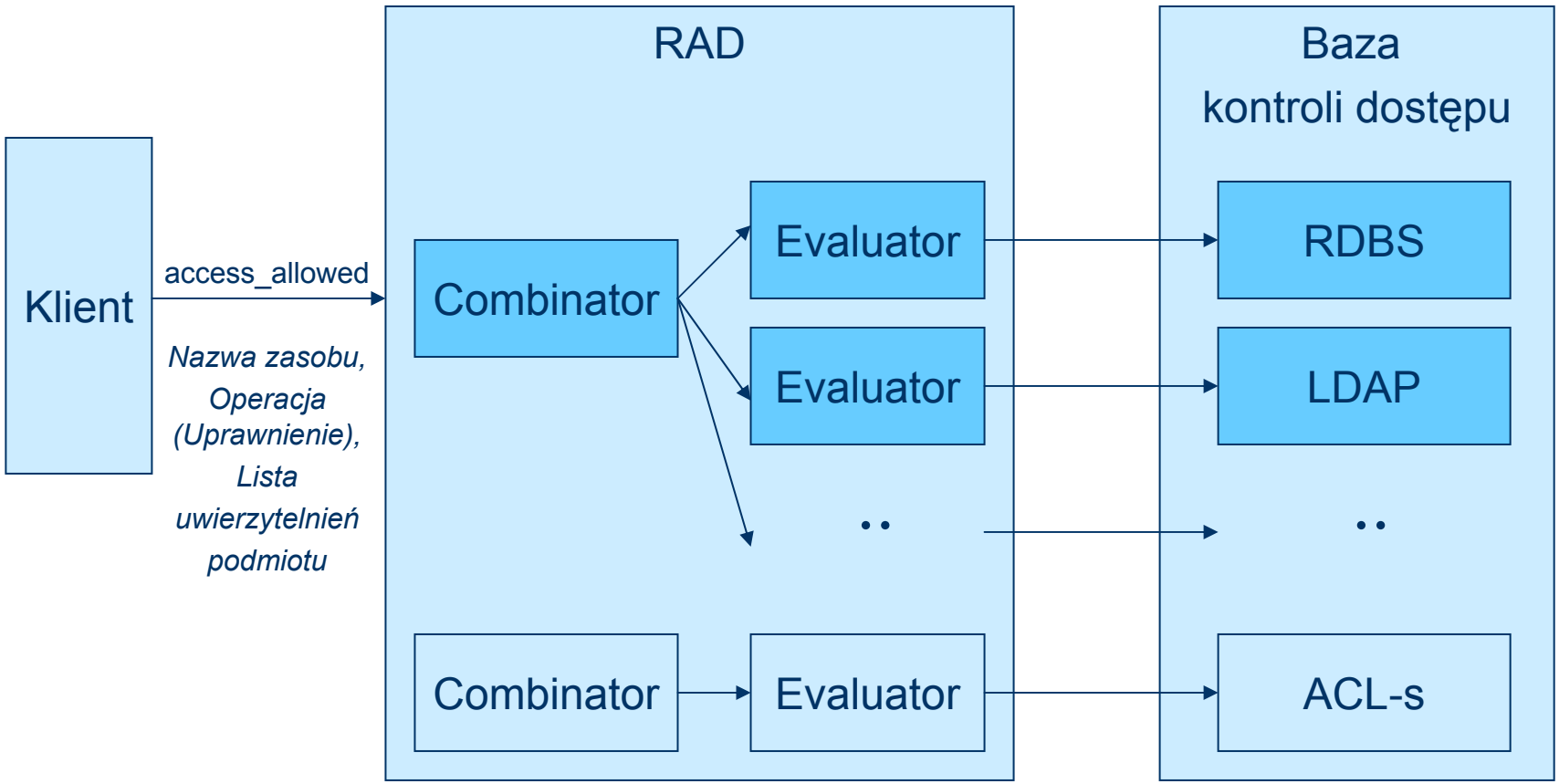
Specyfikacja RAD

- Resource Access Decision Facility Specification
- Object Management Group, Inc. (OMG)
- Kwiecień 2001, wersja 1.0

Architektura RAD-SGI



Przebieg kontroli dostępu



Cel testów

- Zaimplementowana przez nas usługa kontroli dostępu dostarcza interfejsów opartych na:
  - serwer RMI
  - komponenty sesyjne EJB, udostępnione przez RMI lub Web Serwisy
- Celem testów jest zbadanie wydajności usługi kontroli dostępu w zależności od „rodzaju” wykorzystywanego interfejsu

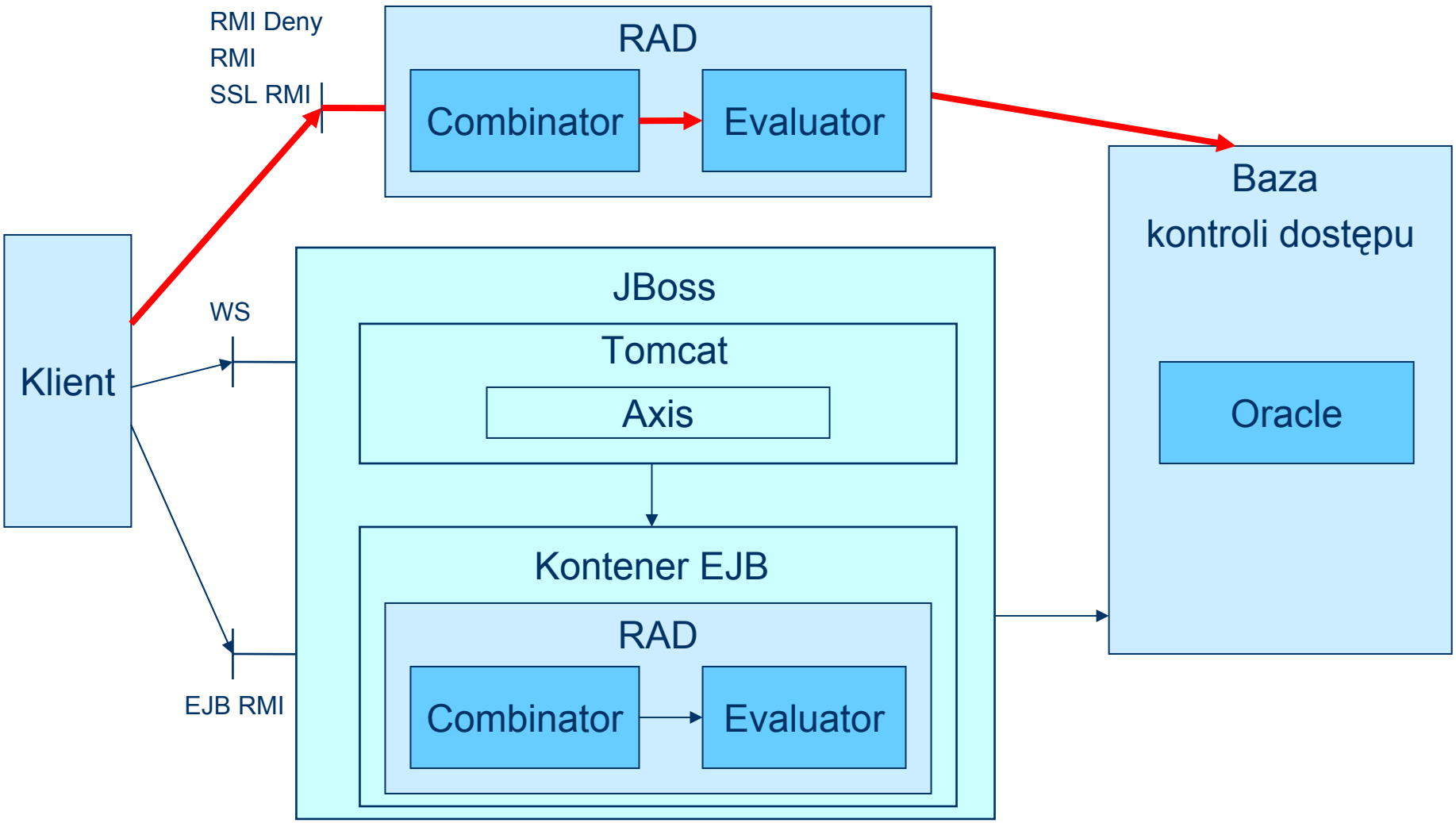
Środowisko testowe

- Sprzęt
  - SUN Midframe 6800, podzielony na 2 domeny:
    (1) 16 procesorów, 16 GB RAM, 100 Mbs
    (2) 8 procesorów, 8 GB RAM, 100 Mbs
- Oprogramowanie
  - Serwer J2EE *JBoss 3.2.3*, baza danych *Oracle 9i*; uruchomione na domenie 1
  - Aplikacja testująca *Grinder*; uruchomiona na domenie 2
  - *J2SE* w wersji 1.4.2_02
  - Aplikacja *jvmstat* służąca do monitorowania *Garbage Collectora*

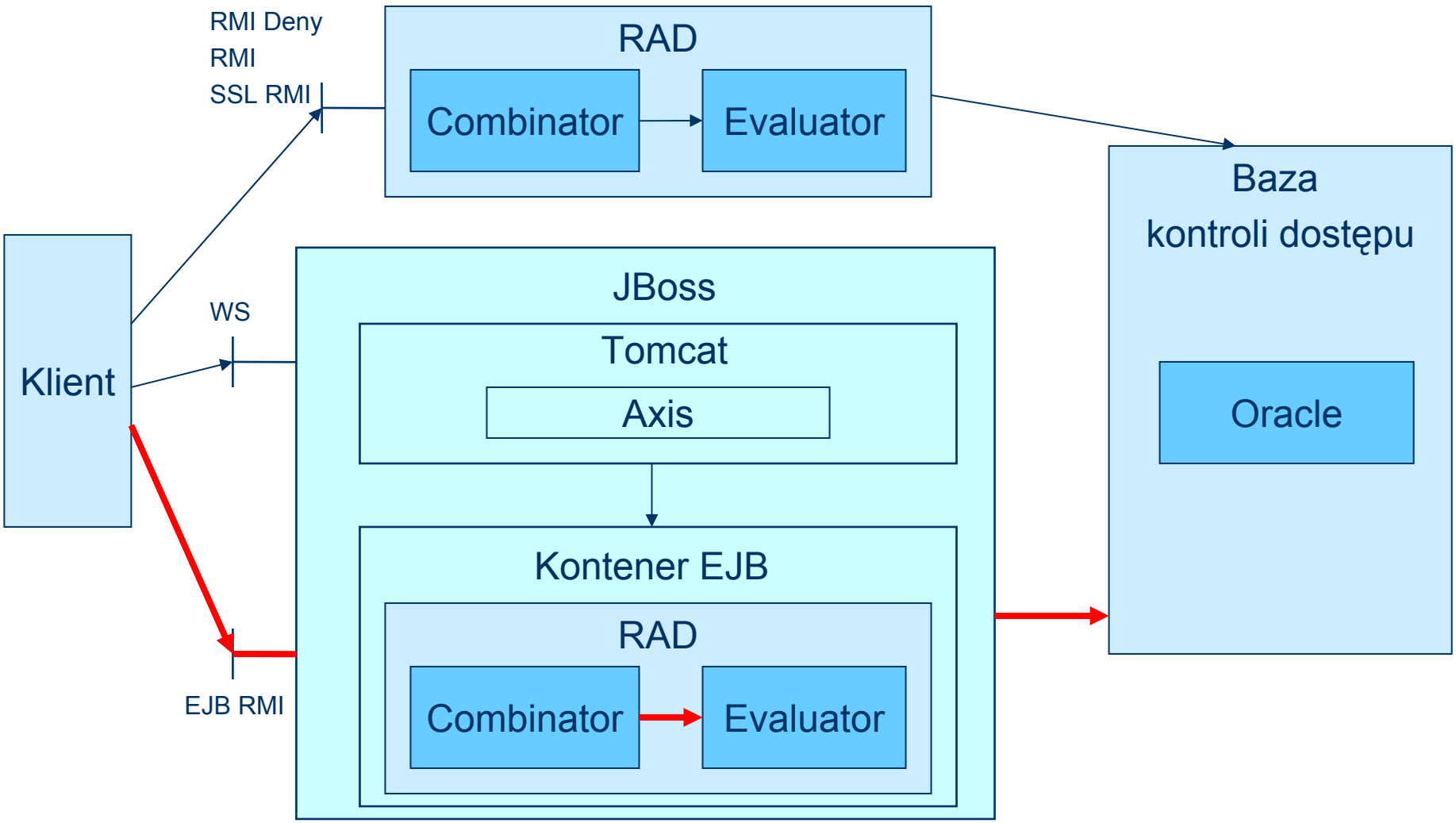
Testowane przypadki użycia

- Zapytania o uprawnienia użytkownika do zasobu
- Klient wykonywał 4 operacje sekwencyjnie z zadanym „*think time*” – 5 sekund
  - 2 operacje zwracały „*true*”, 2 operacje zwracały „*false*”
- Operacje powtarzane 1000 razy
- Różne metody komunikacji
- Zmienna liczba klientów: <1,300>

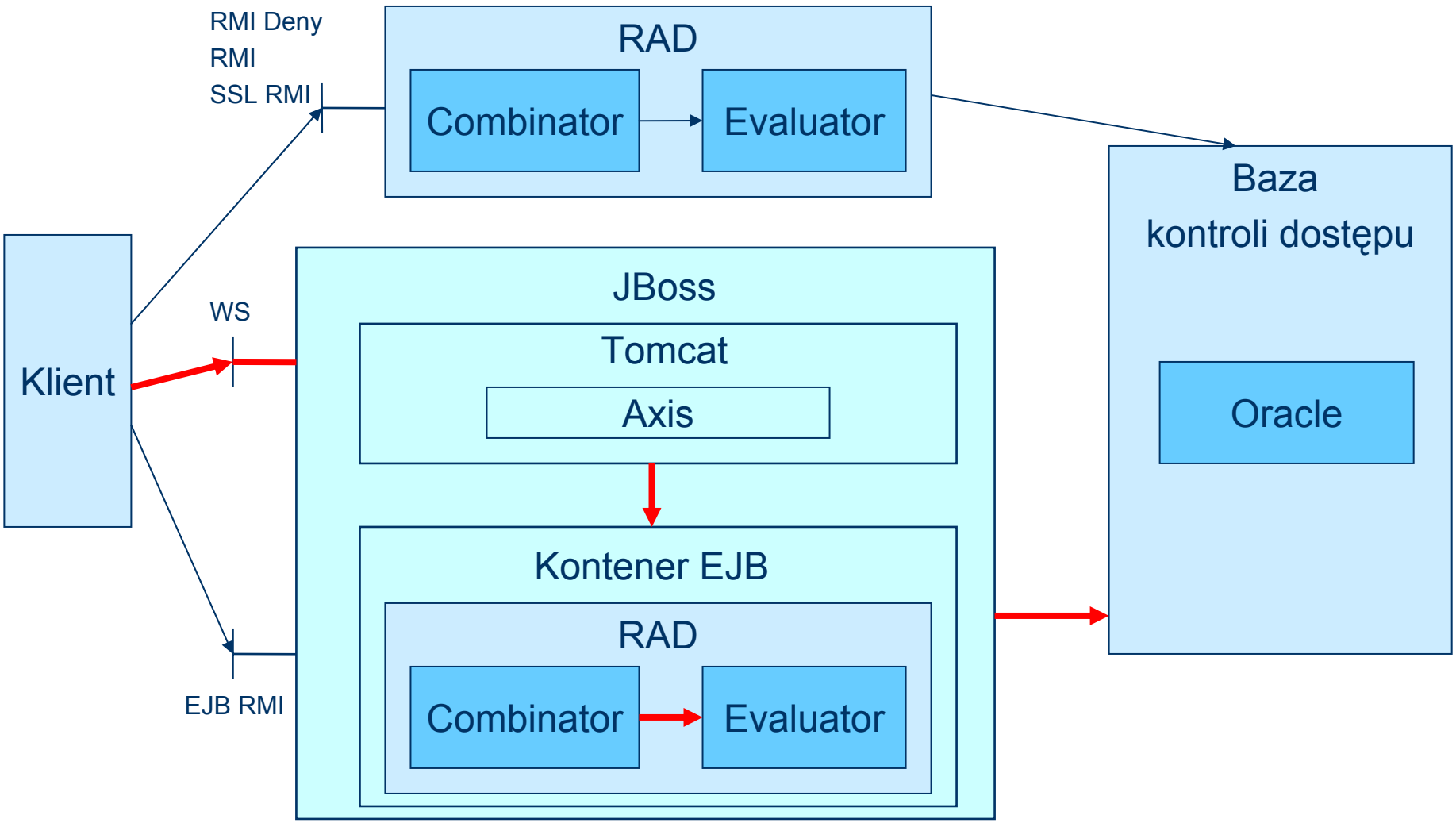
Testowane przypadki użycia ..



Testowane przypadki użycia ..



Testowane przypadki użycia ..



Tuning

- Standardowe ustawienia parametrów wymagają analizy w przypadku gdy chcemy poprawić wydajność
- Serwer aplikacji – pula połączeń (wątki, połączenia do bazy danych), timeouty
- JVM – rozmiar stosu (Xms, Xmx), typ *Garbage Collectora,* sposób optymalizacji *JIT* (server, client) i inne ...

Parametry testów

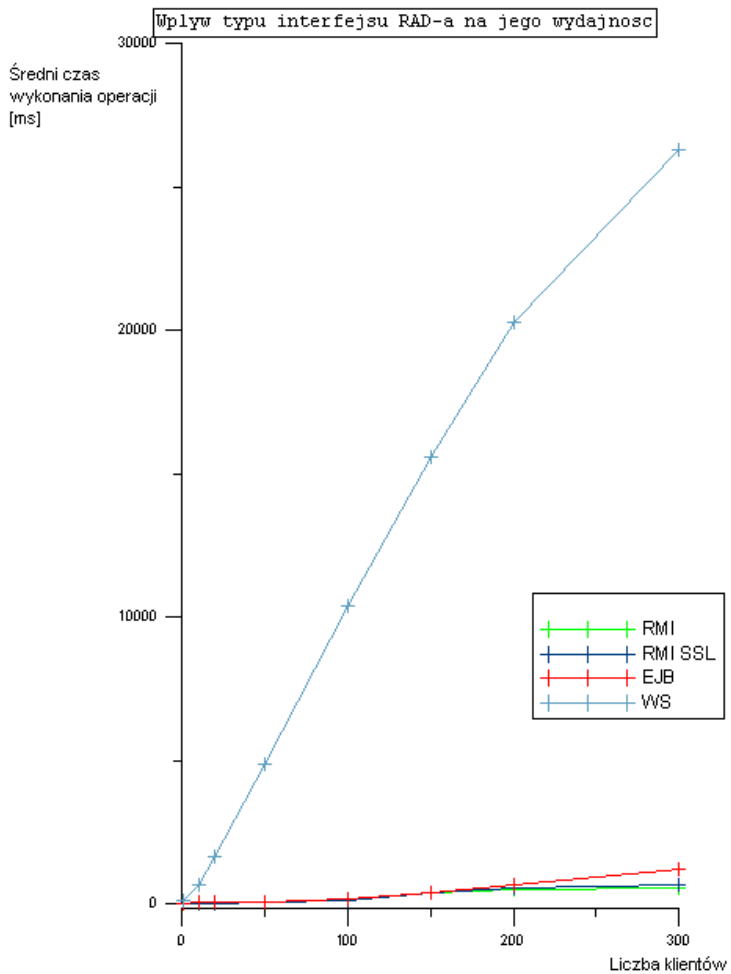
Typ interfejsu	Parametry
RMI/RMI SSL	JRE 1.4.2_02 Xms=256M Xms=256M, -server, standardowy GC JDBC Oracle thin driver, pula 10 połączeń
EJB	JRE 1.4.2_02 Xms=256M Xms=256M, -server, standardowy GC JDBC Oracle thin driver, pula 15 połączeń
WS	JRE 1.4.2_02 Xms=768M Xms=768M, -server, standardowy GC JDBC Oracle thin driver, pula 15 połączeń Tomcat: pula 15 wątków, acceptCount=250

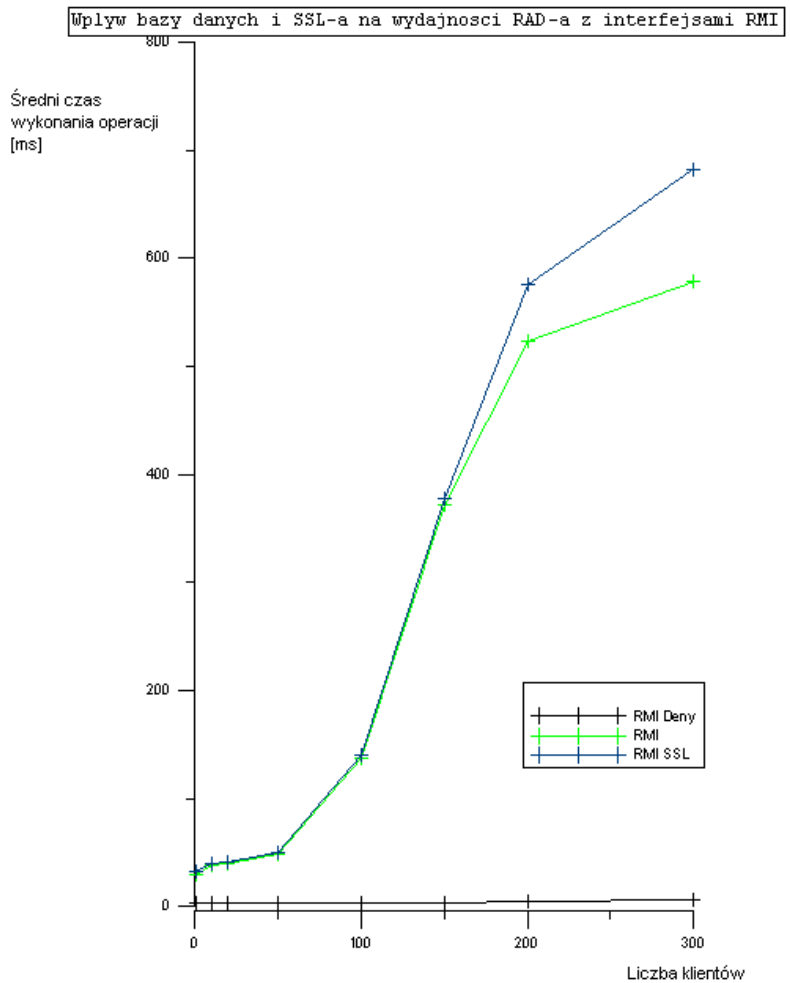
Wyniki

liczba klientów	RMI Deny	RMI	RMI-SSI	EJB	WS
1	3,05	29,5	32,2	37,1	132
10	3,06	38	39,9	51,3	692
20	3,07	39,1	41,6	51,9	1660
50	3,42	49,2	50,3	64,1	4880
100	3,85	138	140	155	10400
150	3,89	372	378	387	15600
200	4,35	524	576	691	20300
300	6,15	578	682	1210	26300

[ms]

Wykresy


Wplyw typu interfejsu RAD-a na jego wydajnosc
Średni czas wykonania operacji [ms]
30000
20000
10000
0
0
100
200
300
Liczba klientów
RMI
RMI SSL
EJB
WS


Wplyw bazy danych i SSL-a na wydajnosci RAD-a z interfejsami RMI
Średni czas wykonania operacji [ms]
800
600
400
200
0
0
100
200
300
Liczba klientów
RMI Deny
RMI
RMI SSL

Konkluzje

- Najlepiej skalowalna okazała się implementacja RAD-a oparta o serwer RMI
- Serwer aplikacyjny JBoss oferuje niewiele mniejszą wydajność w przypadku odwołań RMI do RAD-a w kontenerze EJB
- Web Serwisy ze względu na słabą wydajność mogą mieć tylko ograniczone zastosowanie w przypadku RAD-a