

XV Konferencja PLOUG  
Kościelisko  
Październik 2009

# Współbieżność procesów wsadowych w aplikacjach bazodanowych

Jan Posiadała, Hubert Klekowicz  
Scott Tiger SA

*Jan.Posiadala@tiger.com.pl, Hubert.Klekowicz@tiger.com.pl*

**Abstrakt.** Jedną z podstawowych funkcjonalności systemu bankowego jest przetwarzanie reprezentowanych obiektów w trybie wsadowym. Przetwarzanie złożone z wielu następujących po sobie procesów wsadowych odbywa się codziennie, zaś okresowo (koniec miesiąca, roku, okresu rozliczeniowego) wykonywany jest bardziej rozbudowany zestaw procesów. Oczwistym wymaganiem dla takiej funkcjonalności jest wysoka wydajność przetwarzania.

W pracy przedstawiono modelowe rozwiązanie spełniające wyżej przedstawione wymagania. Zostało ono oparte o system FARMER – implementację ogólnego modelu procesu wsadowego w środowisku bazodanowym. Zaprezentowano wykorzystanie współbieżności (wielowątkowości) przetwarzania wsadowego jako własności zapewniającej wysoką wydajność i skalowalność systemu. Przyjęte rozwiązania technologiczne korzystają wprost z wieloprocesorowej architektury serwerów. Współbieżność dowolnego poziomu (liczba wątków) wynika bezpośrednio z zaimplementowanego ogólnego modelu przetwarzania.

Zaprezentowano wyniki testów wydajnościowych Centralnego Systemu Bankowego ARA, opartego na systemie FARMER, przeprowadzonych w czerwcu 2008 w laboratorium IBM oraz ich interpretację biznesową. Przedyskutowano również inne bezpośrednie korzyści wynikające ze stosowania systemu FARMER w tym m.in. zredukowane kodowanie, łatwiejsze i wydajniejsze testowanie, tańsza optymalizacja oraz utrzymanie systemu.

## 1. Wprowadzenie

W głównej części pracy przedstawiono koncepcję ogólnego modelu procesu wsadowego (OMPW). Rozważania rozpoczęto od sformułowania paradygmatu procesu wsadowego w środowisku bazodanowym. Dalej posiłkując się doświadczeniem w realizacji takich procesów wyspecyfikowano wymagania nałożone na ich realizację tworząc w ten sposób ogólny model procesu wsadowego. Zaprezentowano system FARMER zapewniający szereg funkcjonalności implementacjom procesów zgodnych z OMPW. Jako najważniejszą własność będącą skutkiem zgodności z OMPW przedstawiono przezroczystą współbieżność dla przetwarzania wsadowego.

W części drugiej zaprezentowano wyniki testów procesów wsadowych zrealizowanych w Centralnym Systemie Bankowym ARA. Testy przeprowadzono na wieloprocesorowej maszynie IBM Mainframe z silnym wykorzystaniem współbieżności procesów wsadowych. Szczególną uwagę zwrócono na bezwzględna wydajność oraz skalowalność systemu. W tym kontekście przedstawiono interpretację biznesową wyników testów.

W części trzeciej przedstawiono pozostałe praktyczne korzyści zastosowania systemu FARMER ze szczególnym uwzględnieniem redukcji nakładu pracy w poszczególnych fazach wytwarzania i eksploatacji oprogramowania opartego na procesach wsadowych w środowisku bazodanowym.

## 2. Ogólny model procesu wsadowego

### 2.1. Paradygmat

Z punktu widzenia logiki biznesowej proces wsadowy to proces wykonujący pewne przetwarzanie na zbiorze obiektów reprezentowanych w bazie danych. Na potrzeby niniejszych rozważań przyjęto również, że rozmiar przetwarzanego zbioru jest duży<sup>1</sup>. Zatem, do realizacji tak rozumianej procesu wsadowego trzeba co najmniej reprezentować:

- zbiór przetwarzanych danych
- procedurę przetwarzającą jeden element tego zbioru.

W środowisku bazodanowym jako reprezentację zaproponowano:

- reprezentacją zbioru danych jest widok (predefiniowane zapytanie SQL), wiersze w widoku reprezentują elementy zbioru
- reprezentacją procedury jest procedura składowana z jednym argumentem typu wierszowego – zgodnym typem wiersza widoku

Przy tak przyjętej reprezentacji najprostszą realizacją procesu jest program iterujący w pętli wiersze widoku i aplikujący procedurę do każdego wiersza.

### 2.2. System FARMER

Lata doświadczeń w tworzeniu oprogramowania opartego na bazach danych pokazują, że na realizację tak pojmowanego procesu jest nałożone szereg zapotrzebowań i ograniczeń wynikających ze specyfiki baz danych, specyfiki konkretnych systemów zarządzania bazami danych, potrzeb użytkowników i administratorów. Zatem implementacja procesu w konkretnym języku programowania, w konkretnym systemie zarządzania bazą danych, dla konkretnego klienta musi uwzględniać następujące kwestie:

---

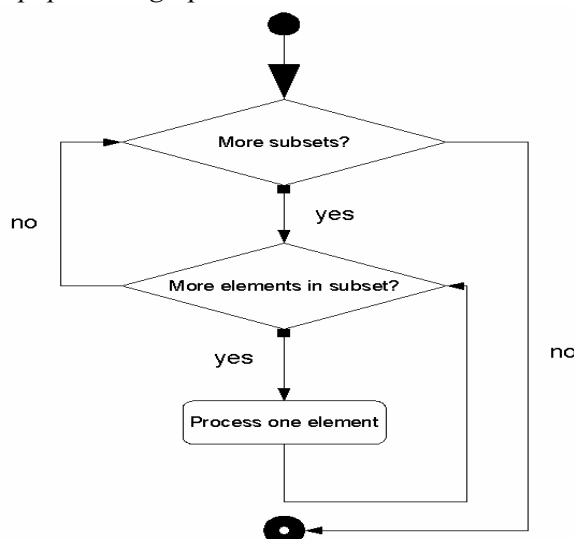
<sup>1</sup> Powyżej 10<sup>4</sup> elementów

- politykę zarządzania transakcjami bazy danych
- dostępność i spójność przetwarzanych danych dla żądań zdalnych
- politykę obsługi błędów
- mechanizm zatrzymywania i wznowiania wykonywanego procesu
- mechanizm kontroli postępu procesu
- optymalizację, diagnostykę, naprawianie błędów
- bezwzględną wydajność przetwarzania
- skalowalność przetwarzania

Niektóre z powyższych ograniczeń spowodowały wypracowanie szeroko stosowanego praktycznego schematu implementacji procesów wsadowych polegającego na podziale przetwarzanego zbioru na podzbiory (najlepiej zbliżonego rozmiaru). Naturalnym sposobem implementacji takiego modelu przetwarzania jest użycie dwóch zagnieżdżonych pętli:

- zewnętrznej – iterującej podział na podzbiory
- wewnętrznej – iterującej elementy podzbioru

Oparcie implementacji o taki schemat pozwala na realizację funkcjonalności uwzględniających większość wyżej wymienionych zapotrzebowań i ograniczeń. Poprawność takiego przetwarzania jest zapewniona przez **prawidłowy podział** przetwarzanego zbioru obiektów na podzbiory. Prawidłowy podział na podzbiory zapewnia, że każdy element zbioru znajduje się w dokładnie jednym z podzbiorów. W praktyce najczęściej każdy element podzbioru ma składową (wartość w kolumnie widoku) będącą (unikalnym) identyfikatorem elementu. W takiej sytuacji podział na podzbiory następuje ze względu na wartość identyfikatora i zachowuje następującą własność **monotoniczności**: wszystkie identyfikatory elementów z *następnego* podzbioru są większe od identyfikatorów elementów z *poprzedniego* podzbioru.



Rys. 1. Zagnieżdżone pętle – schemat przetwarzania

W toku prac rozpoznano dalej idące uogólnienia będące podstawą modelowego rozwiązania pozwalającego na spełnienie omawianych wymagań, co doprowadziło do stworzenia ogólnego modelu procesu wsadowego (OMPW). Model ten zakłada wydzielenie znacznych części funkcjonalności wspólnych dla klasy procesów wsadowych w środowisku bazodanowym – w tym szkieletu przetwarzania opartego na dwóch zagnieżdżonych pętlach.

OMPW wyrażony jest za pomocą specyfikacji wymagań dotyczących interfejsu wystawianego w ramach realizacji procesu wsadowego:

Założenia dotyczące widoku reprezentującego zbiór danych:

Widok ma następujące wyróżnione kolumny sterujące:

- `ID` – będącą kluczem głównym w relacji reprezentowanej przez widok
- `OBJECT_ID` – identyfikator logicznego obiektu reprezentowanego w systemie<sup>2</sup>

Dane udostępniane w widoku są stabilnie i rosnąco posortowane odpowiednio po kolumnach `OBJECT_ID`, `ID`. Takie sortowanie jest jednoznaczne.

W bazie danych istnieją struktury indeksujące pozwalające na szybkie wyszukiwanie zakresowe po wartościach w kolumnach `ID` oraz `OBJECT_ID`.

Dodatkowo zaleca się, aby procedura przetwarzająca wiersz widoku nie odwoływała się do kolumn `ID` oraz `OBJECT_ID`. W razie potrzeby użycia tych wartości należy użyć kolumny o niezastereżonej nazwie.

Implementacja procesu udostępnia jednoparametrową procedurę `one`. Parametr jest zgodny z typem wierszowym widoku reprezentującego zbiór danych.

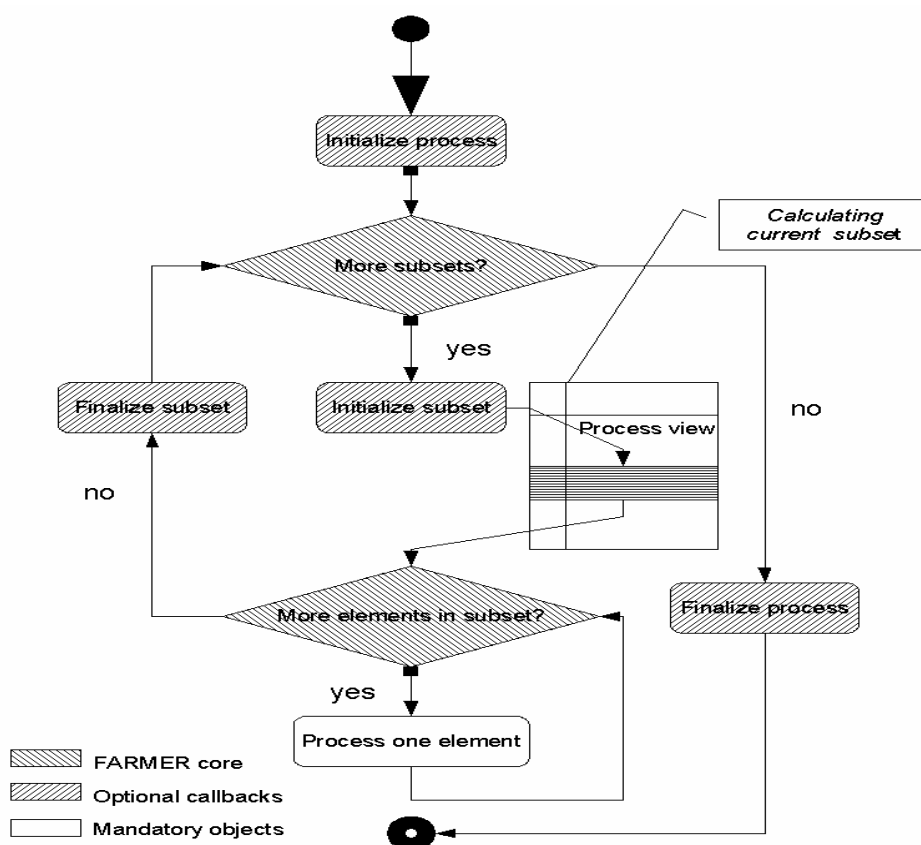
Dodatkowo implementacja procesu może udostępniać następujące wywołania zwrotne:

- `init_process` – będzie wykonywana przed wejściem do zewnętrznej pętli
- `init_subset` – będzie wykonana przed każdorazowym wejściem do pętli wewnętrznej
- `finalize_subset` – będzie wykonana po każdorazowym wyjściu z pętli wewnętrznej
- `finalize_process` – będzie wykonana po wyjściu z pętli zewnętrznej

Powyższa specyfikacja bazuje na obserwacji, że wykonanie procesu przebiega według zilustrowanego poniżej schematu:

---

<sup>2</sup> Dokładniejsze uzasadnienie wprowadzenia tej kolumny podamy w części dotyczącej realizacji współbieżności



Rys. 2. Ogólny model procesu wsadowego – schemat przetwarzania

W przypadku skrajnym (ale nierzadkim) implementacja procesu jest bezpośrednią realizacją paradygmatu procesu wsadowego. Oznacza to, że do realizacji procesu wsadowego potrzeba i wystarcza reprezentować zbiór przetwarzanych obiektów i procedurę przetwarzającą jeden obiekt.

### 2.3. Realizacja współbieżności

Na potrzeby realizacji współbieżności wprowadzono intuicyjne założenie dotyczące kolumny `OBJECT_ID`.

Przetwarzanie elementów o różnych wartościach `OBJECT_ID` jest logicznie niezależne (mogą być przetwarzane współbieżnie).

Realizacja współbieżności w skonstruowanym dotąd OMPW polega na takiej synchronizacji i komunikacji wątków wykonawczych danego procesu, która skutkuje poprawnym przydziałem kolejnego podzbioru elementów do konkretnego wątku.

Przyjmijmy, że podział na podzbiory odbywa się względem kolumny `OBJECT_ID` oraz że przydział podzbioru elementów do wątku wykonawczego podlega wykonaniu sekwencyjnemu w ramach wszystkich wątków. Wobec tego można stwierdzić, że **prawidłowy i monotoniczny** podział wyjściowego zbioru na podzbiory jest trywialny. Takim podziałem jest dowolny podział zgodnie z porządkiem sortowania nierozdzielający elementów o tym samym `OBJECT_ID` do dwóch podzbiorów.

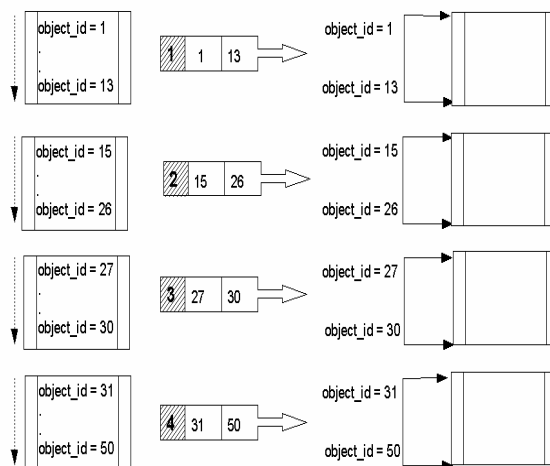
Dla zapewnienia sekwencyjnego wykonania przydziału podzbioru elementów do wątku wykonawczego konieczne jest, aby najpóźniej po zakończeniu wykonania procedury `init_subset` znana była wartość największego identyfikatora `OBJECT_ID` z wyliczonego właśnie podzbioru. Po-

nieważ procedura `init_subset` potrzebuje największego `OBJECT_ID` z poprzednio wyliczonych podzbiorów, procedura `init_subset` podlega wykonaniu w sekcji krytycznej w ramach wielowątkowego wykonania procesu.

Ze względu na brak dodatkowych założeń dotyczących reprezentacji przetwarzanego zbioru i procedury przetwarzającej, realizacja współbieżności jest przezroczysta.

### 2.3.1. Histogramy

Histogramy w `FARMERZE` są strukturami przechowującymi informację o podziale danego zbioru identyfikatorów na podzbiory. W praktyce przechowywane są graniczne identyfikatory podzbiorów, zatem podział jest zgodny porządkiem sortowania. Zastosowanie histogramów pozwala na wyjęcie procedury `init_subset` z sekcji krytycznej. W zamian w sekcji krytycznej wykonywany jest odczyt z histogramu. Jednym kosztem wprowadzenia histogramów jest rezygnacja z dokładnego, ze względu na licznosc, podziału na podzbiory, co pozostaje bez większego wpływu na działanie systemu.



Rys. 3. Histogramy

## 3. Testy wydajnościowe systemu ARA

W czerwcu 2008 r. w ośrodku IBM w Boeblingen odbyły się testy Centralnego Systemu Bankowego ARA autorstwa firmy Scott Tiger S.A. na platformie z/Linux działającej na maszynie z10. CSB ARA jest w pełni funkcjonalnym systemem obsługującym szereg usług bankowych poczynając od rejestracji klienta i rachunku bankowego (bieżącego, oszczędnościowego, kredytowego) poprzez księgowanie transakcji, aż do generowania wyciągu i przygotowywania raportów na potrzeby podmiotów zewnętrznych.

Celem testów było potwierdzenie możliwości wdrożenia CSB ARA na platformie klasy IBM Mainframe jako centralnego systemu bankowego w średniej wielkości banku detalicznym. Testy przeprowadzono ze szczególnym uwzględnieniem aspektów wydajnościowych.

Cel testów został podzielony na następujące składowe:

- i. Potwierdzenie technologicznej stabilności rozwiązania z użyciem:
  - Centralny System Bankowy ARA
  - RDBMS Oracle 10g
  - system operacyjny Linux

- platforma sprzętowa IBM System z10
- ii. Zbadanie bezwzględnej wydajności systemu CBS ARA poprzez maksymalne obciążenie jednostek obliczeniowych i urządzeń pamięci masowej
- iii. Zbadanie skalowalności systemu poprzez uruchamianie testowanych funkcjonalności na liczbie wątków zależnej od sprzętowej konfiguracji serwera centralnego

### 3.1. Środowisko testowe

#### 3.1.1. Środowisko sprzętowe

Środowisko testowe zbudowane zostało w oparciu o 3 główne elementy:

- IBM Mainframe z10
- serwery Blade
- macierz dyskowa DS8300

Serwery kasetowe Blade w liczbie 10 posłużyły jako generatory obciążenia dla testowego systemu ARA. Podsystem dyskowy stanowiła macierz DS8300 wyposażona w 11 dysków.

#### 3.1.2. System operacyjny

Testy były prowadzone w obrębie dedykowanego LPAR'u wewnątrz którego zainstalowano system operacyjny Suse Linux Enterprise Server 10 SP2 (SLES 10 SP2). Konfiguracja sprzętowa LPAR'u była zmieniana celem przeprowadzenia testów na różnej liczbie procesorów i pamięci wg poniższej tabeli:

Tabela. 1. Zasoby sprzętowe w konfiguracjach testowych

Konfiguracja	Liczba procesorów	Pamięć RAM
P2	2	16 GB
P4	4	16 GB
P8	8	64 GB

#### 3.1.3. Baza danych

Na potrzeby testów przygotowano instancję i bazę danych Oracle 10g w następującej konfiguracji:

Oracle Database 10g Enterprise Edition Release 10.2.0.3.0 – 64bit Production  
With the Partitioning, OLAP and Data Mining options

Testowany system zasillono danymi stanowiącymi odpowiednik zawartości produkcyjnej bazy danych:

- 7 mln rachunków bieżących
- 3 mln kart kredytowych

Baza danych przed rozpoczęciem testów zawierała łącznie 472 mln rekordów we wszystkich tabelach aplikacyjnych. Po zakończeniu było ich 1950 mln.

### 3.1.4. Procedury testowe

Podczas testów wykonano następujące zestawy testów:

iv. Zestaw A – symulacja obciążenia systemu operacjami on-line

v. Zestaw B – symulacja przetwarzania wsadowego w module kart kredytowych

- B1. księgowanie operacji
- B2. naliczanie odsetek
- B3. księgowanie odsetek

Zestaw C – symulacja przetwarzania wsadowego w module rachunków bieżących

- C1. księgowanie operacji
- C2. naliczanie odsetek
- C3. kapitalizacja odsetek

## 3.2. Wyniki testów

### 3.2.1. Wydajność procesów wsadowych

Na potrzeby testów procesów wsadowych określono następujące miary wydajności:

- TPS – średnia liczba operacji przetwarzanych w czasie jednej sekundy dla testów B1, C1
- PPS – średnia liczba produktów przetwarzanych w czasie jednej sekundy dla testów B2, B3, C2, C3

W tabelce przedstawiono wyniki testów procesów wsadowych dla poszczególnych konfiguracji zasobów sprzętowych.

Tabela. 2. Wyniki testów procesów wsadowych

Test	Konfiguracja P2		Konfiguracja P4		Konfiguracja P8		Współczynnik współbieżności
	Wydajność	Liczba wątków	Wydajność	Liczba wątków	Wydajność	Liczba wątków	
B1 (TPS)	2304	4	4600	8	8631	16	2
B2 (PPS)	2175	4	4267	8	8264	16	2
B3 (PPS)	1811	3	3304	6	5703	12	1.5
C1 (TPS)	1033	3	1754	6	2511	12	1.5
C2 (PPS)	7143	5	13605	10	25806	20	2.5
C3 (PPS)	999	3	1927	6	3876	12	1.5

W kolumnie „Liczba wątków” zaprezentowano liczbę wątków, przy której uzyskano najlepszą wydajność dla danego testu w danej konfiguracji. **Współczynnik współbieżności** dla danego testu zdefiniowano jako stosunek liczby wątków do liczby procesorów w konfiguracji sprzętowej. Wiadac, że optymalny współczynnik współbieżności oscyluje wokół 2 ( $\pm 0.5$ ).

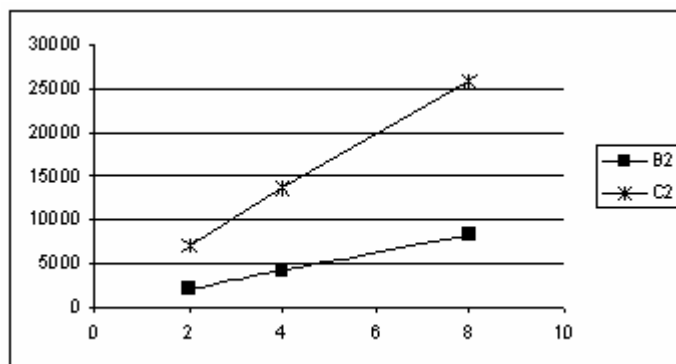
### 3.2.2. Skalowalność procesów wsadowych

Testowane procesy przy zmianie konfiguracji zachowywały dobrą liniową skalowalność pionową. Średnio dla testowanych procesów współczynnik skalowalności przy 2-krotnym rozszerzeniu konfiguracji wynosił powyżej 0.9.

Tabela. 3. Skalowalność pionowa przetwarzania wsadowego

P2→P4	P4→P8	P2→P8
0.94	0.90	0.85

Poniżej wykres przedstawiający skalowalność dla procesów naliczania odsetek: dla rachunków bieżących i kart kredytowych.



Wykres. 1. Skalowalność pionowa procesów naliczania odsetek

### 3.3. Interpretacja biznesowa wyników testów

#### 3.3.1. Charakterystyka klientów

Przyjęto, iż każdy klient posiada 2 aktywne produkty bankowe:

- każdy klient posiada rachunek bieżący
- każdy klient posiada albo lokatę, albo kartę kredytową

Zatem np. 10 mln klientów posiada 20 mln aktywnych produktów bankowych. Przyjęto ponadto, że generowanie wyciągów dla kart kredytowych rozłożone jest na 10 cykli bilingowych (CB).

#### 3.3.2. Przetwarzanie miesiąca

Przyjęto, że na koniec miesiąca kalendarzowego (zarazem dzień cyklu bilingowego) wykonywany jest następujący zestaw procesów wsadowych:

- dla rachunków (zamknięcie miesiąca)
  - i. 1 wykonanie procesu typu C1
  - ii. 1 wykonanie procesu typu C2
  - iii. 1 wykonanie procesu typu C3
- dla kart kredytowych (generacja wyciągów dla 1 CB)
  - i. 3 wykonania procesu typu B1 (dla kart z jednego CB)
  - ii. 1 wykonanie procesu typu B2 (dla wszystkich kart)
  - iii. 3 wykonanie procesu typu B3 (dla kart z jednego CB)
- dla lokat (zamknięcie miesiąca)
  - i. 1 wykonanie procesu typu C1
  - ii. 1 wykonanie procesu typu C2

Ograniczenie czasowe: przyjęto, że przetwarzanie miesiąca może trwać maksymalnie 180 minut.

### 3.3.3. Przetwarzanie miesiąca

Przyjęto, że podczas przetwarzania zamknięcie dnia (zwykłego) wykonywany jest następujący zestaw procesów wsadowych:

- dla rachunków
  - i. 1 wykonanie procesu typu C2
- dla kart kredytowych (generacja wyciągów dla 1 CB)
  - i. 1 wykonania procesu typu B1 (dla kart z jednego CB)
  - ii. 1 wykonanie procesu typu B2 (dla wszystkich kart)
- dla lokat (zamknięcie miesiąca)
  - i. 1 wykonanie procesu typu C2

Ograniczenie czasowe: przyjęto, że przetwarzanie dnia może trwać maksymalnie 40 minut.

Na podstawie przyjętych założeń i wyników testów obliczono maksymalną liczbę klientów możliwych do obsługi w systemie ARA w poszczególnych konfiguracjach. Uzyskane rezultaty zostały zebrane w poniższej tabeli.<sup>3</sup>

Tabela. 3. Interpretacja biznesowa wyników testów

Konfiguracja	Maksymalna liczba klientów (mln)
P2	3.2
P4	6.3
P8	10.9

Wszystkie procesy wsadowe w CBS ARA są zrealizowane w systemie FARMER.

## 4. Korzyści z zastosowania systemu FARMER

### 4.1. Projektowanie

OMPW jest wzbogacony o grupowanie procesów ze względu na własności przetwarzania. Wysszczególniono następujące typy procesów.

- przyrostowy – importy/exparty z/do zdalnych systemów
- cykliczny – przetwarzania okresowe (dziennie, miesięczne, roczne itp.)
- samosterujący – kolejki (zadaniem przetwarzania jest „usunięcie” elementu ze zbioru do przetworzenia)

Powyższe rozróżnienie pozwala na określenie własności procesu we wczesnym etapie realizacji, ułatwia prototypowanie oraz definicję reprezentacji przetwarzanego zbioru.

<sup>3</sup> Podczas obliczania maksymalnej liczby klientów uwzględniono także wyniki testów dla operacji on-line. Dla dwóch konfiguracji okazały się one nieznacznie silniejszym ograniczeniem na maksymalną liczbę klientów. Dla P4 – 5.9 mln, dla P8 – 10.8 mln klientów.

## 4.2. Kodowanie

Jak pokazaliśmy do implementacji standardowego procesu wsadowego wystarczy zaimplementować widok procesowy oraz jednoparametrową procedurę przetwarzającą obiekt reprezentowany przez wiersz widoku. Zatem kodowanie zostaje zredukowane w następujących obszarach:

- iteracja przetwarzanego zbioru
- zarządzanie transakcjami
- obsługa wyjątków na najwyższym poziomie

## 4.3. Testowanie

Realizacja procesu wsadowego w systemie FARMER umożliwia testy jednostkowe przetwarzania przy pomocy standardowych skryptów:

```
BEGIN
  FOR R IN (SELECT *
            FROM V_PROCES_KOSCIELISKI
            WHERE ID IN (1))
  LOOP
    PROCES_KOSCIELISKI.ONE(R);
  END LOOP;
END;
```

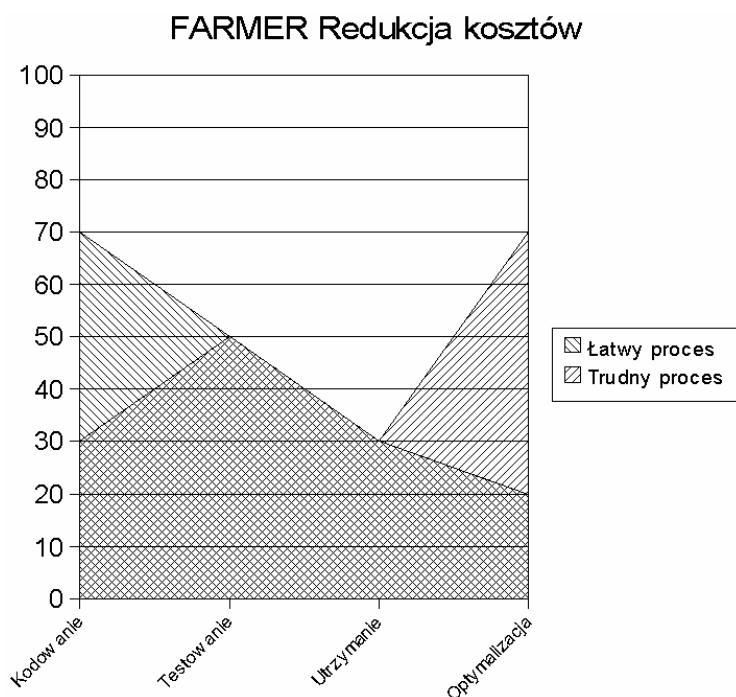
Ponadto w przypadku wdrażania większych aplikacji – takich jak na przykład moduły produktowe CBS ARA – funkcjonalne testy systemowe wymagają weryfikacji działania (w tym współdziałania) większej ilości procesów wsadowych. W takich aplikacjach najczęściej widoki procesowe oparte są o wspólny widok zawierający wszystkie obiekty zarządzane przez daną aplikację. Na potrzeby testów zbiór obiektów prezentowanych przez wspólny widok może zostać w prosty sposób ograniczony do zadanego zestawu testowego.

## 4.4. Utrzymanie i optymalizacja

System FARMER zapewnia wspólną dla procesów wsadowych funkcjonalność ułatwiającą utrzymanie i optymalizację aplikacji:

- zatrzymania procesu na żądanie
- wznowienie procesu w punkcie zatrzymania
- logowanie systemowych komunikatów diagnostycznych o rozpoczęciu, zakończeniu, przerwaniu, wznowieniu wykonywania procesu
- logowanie komunikatów o błędach
- przechowywanie informacji o wydajności procesu
- przechowywanie informacji o zużyciu zasobów sprzętowych przez wykonanie procesu
- przechowywanie informacji o zrównoważeniu rozkładu pracy pomiędzy wątkami
- przechowywanie informacji o liczbie i długości wystąpień zdarzeń oczekiwania (*wait event*) – w tym zdarzeń z klas *Application*, *User I/O*, *Concurrency*
- przechowywanie informacji o planach i statystykach wykonań poleceń SQL (*SQL trace*)

Szacuje się, że redukcja nakładu pracy w poszczególnych fazach wytwarzania oprogramowania (kodowania, testowania, utrzymania, optymalizacji) przy zastosowaniu systemu FARMER waha się od 20% do 70% w zależności od stopnia skomplikowania procesu.



Wykres. 2. Redukcja nakładu pracy przy zastosowaniu systemu FARMER

## 5. Podsumowanie

W pracy omówiono modelowe podejście do zagadnienia procesów wsadowych w środowisku bazodanowym. Sformułowano paradygmat, który wraz z wiedzą wynikającą z obserwacji stał się podstawą opracowania ogólnego modelu procesu wsadowego (OMPW). Przedstawiono specyfikację zgodności z OMPW oraz założenia systemu FARMER wykorzystującego OMPW do realizacji procesów wsadowych.

Szczególną uwagę zwrócono na możliwość przezroczystej realizacji współbieżności jako własności bezpośrednio wynikającej z Ogólnego Modelu oraz zaawansowany sposób wykorzystującej możliwości nowoczesnych maszyn wieloprocesorowych. W tym kontekście zaprezentowano wyniki testów systemu bankowego ARA na IBM Mainframe z10. Uzyskane wyniki potwierdziły możliwość zastosowania systemu ARA na maszynie IBM Mainframe jako centralnego systemu bankowego na rynku polskim.

Omówiono wpływ użycia systemu FARMER na redukcję kosztów produkcji i eksploatacji oprogramowania bazującego na przetwarzaniu wsadowym w środowisku bazodanowym.

## Bibliografia

- [GiPo08] Gilarski K., Posiadała J.: Central Banking System ARA. Portability and performance tests on IBM Mainframe platform. Test plan., 2008
- [KMPP08] Klekowicz H., Michalak P., Pietrzak P., Posiadała J.: Centralny System Bankowy ARA. Testy rozwiązania na maszynie klasy IBM Mainframe. Raport., 2008