STUDIA INFORMATICA 2008
Volume 29 Number 1 (75)

Marcin GORAWSKI, Wojciech KOŁODZIEJ
Politechnika Śląska, Instytut Informatyki

ZNACZNIE ROZPROSZONA SIEĆ PRZETWARZANIA STRUMIENIOWEGO

Streszczenie. W celu zasilania rozproszonych systemów analitycznych z rozproszonych systemów źródłowych zaproponowano rozproszony system ekstrakcji strumieni danych, oparty na sieci inżynierów procesu ETL. Taka sieć ETLPE, oparta na mechanizmie rozproszonych zasobów, pozwala na automatyzację projektowania znacznie rozproszonego strumieniowego procesu ETL i zapewnienie pełnej jego kontroli z dowolnego miejsca.

Słowa kluczowe: hurtownie danych, ekstrakcja rozproszonego strumienia danych, silniki strumieniowe

WIDELY DISTRIBUTED STREAM PROCESSING NETWORK

Summary. In order to feed widely distributed analytical systems with data from widely distributed source systems, a widely distributed stream data extraction system was proposed. The system bases on ETL process engineers. Such network, the ETLPE, based on a distributed resources mechanism, allows automation of the widely distributed stream process design and ensures fully remote online control.

Keywords: data warehouse, distributed stream data extraction, stream engine

1. Wprowadzenie

W miarę upływu czasu uwidacznia się zapotrzebowanie na przetwarzanie zapytań w znacznie rozproszonych systemach strumieniowych. Stosowane metody przetwarzania zapytań zwykle adresowane są do systemów o stosunkowo małej skali rozproszenia, w wyniku czego w rozleglejszych sieciach (np. Internet) przepustowość połączeń znacznie obniża ich wydajność. Obecnie odchodzi się od centralnego modelu zarządzania i systemy wspomagające podejmowanie decyzji (ang. *Decision Support Systems, zwane dalejDSS*) ewoluują w kierun-

ku decentralizacji, a co za tym idzie, rozproszenia ich węzłów nawet na znaczne odległości geograficzne. Stąd problem zasilania systemów DSS aktualnymi danymi w sposób bezpieczny i bezstratny. Równie ważnym zagadnieniem stał się proces centralizacji rozproszonych danych z wszelkiego rodzaju mierników telemetrycznych, detektorów i urządzeń sensorowych.

Prezentujemy **znacznie rozproszoną sieć przetwarzania strumieniowego** złożoną z niezależnych jednostek (węzłów) przetwarzania o nazwie **Inżynierów Procesu ETL** (ang. *ETL Process Engineers* zwane dalej ***ETLPE***), wspomagających proces projektowania, wykonywania i kontroli sieci przepływu strumienia krotek zasilających (sieć ETLPE). W sieci ETLPE zaproponowane zostały interfejsy pozwalające na przeprowadzenie ekstrakcji i zasilania z dowolnego źródła i dowolnego odbiornika, których dane można przedstawić w postaci krotek. Ponadto, całość zadań związanych z ustalaniem źródła i odbiorników krotek można przeprowadzić i kontrolować z dowolnego miejsca sieci ETLPE.

W niniejszym artykule przedstawiamy proces projektowania sieci niezależny od fizycznej topologii sieciowej oraz lokalizacji początkowych i końcowych zasobów procesu (takich jak bazy danych, pliki płaskie, inne źródła i odbiorniki krotek) i zaprezentować ujednolicony interfejs projektowania uruchamiania i monitorowania procesu.

W pierwszej części artykułu zostaną przedstawione prace pokrewne (rozdział 2.), następnie zaprezentujemy koncepcję i budowę sieci Inżynierów Procesu ETL (rozdział 3.). W rozdziale 4 zasygnalizujemy przyszłe plany rozwoju i podsumujemy dotychczasową pracę.

2. Prace pokrewne

Nasza praca nad siecią ETLPE nawiązuje do wielu wcześniejszych badań, dotyczących zarządzania danymi, silników strumieniowych oraz inteligentnej automatyzacji procesu.

Silniki strumieniowe. Zagadnienia i projekty prototypowych silników strumieni danych są intensywnie badane w pracy *Borealis* i jej poprzednikach [1]. *Borealis* jest rozproszonym silnikiem przetwarzania strumieni drugiej generacji rozwijanym na Uniwersytetach: Brandeis, Browna oraz MIT. *Borealis* dziedziczy podstawową funkcjonalność strumieniową po silniku *Aurora*, a funkcjonalność rozproszoną po silniku *Medusa*. Rozwój tych technologii idzie w kierunku bezpieczeństwa i niezawodności transmisji oraz rozproszonej, aktywnej optymalizacji zapytań. W systemie *Borealis*, podobnie jak i w naszej sieci ETLPE, zdefiniowany jest optymalizator, który kontroluje proces zarządzania zasobami i może wpływać na jego przebieg. Systemy typu *Borealis* działają na zdefiniowanych zasobach. W sieci ETLPE rozważamy system globalnych zasobów oraz większą decentralizację procesu – każda jednostka może przejąć funkcję kontrolną. Systemy typu *Borealis* nie

rozpatrują optymalizacji pod kątem znacznie rozproszonych systemów, gdzie decydujące znaczenie ma przepustowość i obciążenie sieciowe. W naszym rozwiązaniu minimalizujemy obciążenie transmisji krotek w takich systemach.

Dynamiczna dystrybucja obciążenia. Rozproszone systemy strumieniowe muszą być odporne na nieprzewidywalne zmiany obciążenia strumienia i odpowiednio do potrzeb alokować dostępne zasoby. Jedną z ciekawszych prac w tym kierunku jest [2]. Autorzy tej pracy proponują inteligentny rozproszony system oparty na serwerach *Tapestry*, pozwalający na świadomą alokację operatorów w zależności od obciążenia sieci i jednostek przetwarzania. W odniesieniu do tego rozwiązania proponujemy alternatywny model równorzędnych inżynierów procesu z rozproszonym globalnym systemem zasobów i rozproszonym systemem kontrolno-sterującym.

Strumieniowy system ETL tolerujący błędy. W pracy [3] przedstawiono system zapewniający ciągłość procesu ETL, pobierającego dane ze źródeł strumieniowych, lub o ile to możliwe, powrót do stanu normalnej pracy przerwanego przetwarzania, bez żadnych strat danych, bez względu na czas trwania awarii. Zaprojektowany system jest zestawem komunikujących się ze sobą modułów o różnej funkcjonalności i rozproszeniu geograficznym. System składa się z następujących elementów: a) źródeł strumieni danych (tj. serwerów telemetrycznych lub węzłów zbiorczych), b)modułów buforujących (RBF), c) trwałych integrattorów strumieni (RIF), d) modułów ETL oraz e) modułów wykrywania błędów i integracji strumieni (FTI). Propozycja przedstawiona w niniejszym artykule rozszerza tę idę o możliwość projektowania warstwy ETL na niższych warstwach przetwarzania oraz zapewnia wydajną i bardziej elastyczną strukturę procesu zdolną do rozbudowy i konserwacji.

3. Projekt sieci ETLPE

Projektem sieci ETLPE nazywać będziemy projekt polegający na ustanowieniu zbioru dobranych serwerów ETLPE, przeznaczonych do realizacji procesu ETL i jego kontroli oraz zbioru zadań i instrukcji tworzenia lokalnych grafów ekstrakcji, a także definicji połączeń pomiędzy nimi.

Projektem sieci ETLPE nazywać będziemy zbiór wyselekcjonowanych serwerów ETLPE przeznaczonych do kontroli i wykonania procesu ETL oraz zbiór zadań, zawierający szczegółowe instrukcje utworzenia lokalnych grafów ekstrakcji, a także definicji połączeń pomiędzy nimi.

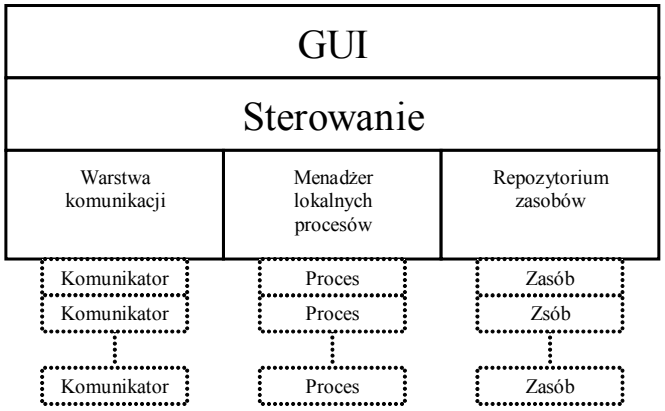
Projekt sieci ETLPE oparty jest na lokalnych repozytoriach reprezentantów zasobów, będących interfejsami do lokalnie zdefiniowanych źródeł lub odbiorników krotek. Zadaniem administratora jest zdefiniowanie pliku konfiguracyjnego reprezentanta zasobu oraz jego

rejestracja na lokalnej jednostce ETLPE (zamiennie zwanej: inżynierem). Taki zasób od momentu rejestracji staje się widoczny dla wszystkich podłączonych jednostek ETLPE i może uczestniczyć w procesie tworzenia grafu ETL.

Jednym z kluczowych założeń projektu jest zaprezentowanie użytkownikowi wszystkich globalnie zdefiniowanych zasobów dla dowolnej jednostki ETLPE (bez zagłębiania się w strukturę połączeń i lokalizację każdego z inżynierów oraz udostępnienie mu odpowiednich mechanizmów wymaganych jedynie do wybrania zasobu zasilającego i zasobów zasilanych).

Fizyczny graf ekstrakcji ETL budowany jest za pomocą optymalizatora projektu, który na podstawie odległości sieciowych pomiędzy jednostkami sąsiadujących inżynierów oraz zdefiniowanych reprezentantów zasobów typuje najlepszą drogę dla procesu oraz definiuje odpowiednie zadania dla każdej pośredniczącej w tym procesie jednostki przetwórczej.

Schemat jednostki ETLPE przedstawiony jest na rysunku 1. Schemat sieci ETLPE wraz z zarejestrowanymi zasobami zaprezentowany został na rysunku 2. Każda jednostka ETLPE jest niezależna i może być dowolnie łączona z inną jednostką sieci ETLPE, pod warunkiem że obie sieci są wzajemnie widoczne – mogą zdefiniować dwustronne aktywne połączenie. Każdy ETLPE włączony do globalnej sieci ETLPE ma jednakowe prawa i niezależnie od miejsca i struktury sieci połączeń jednakową widoczność (dotyczy to przede wszystkim zasobów).



Rys . 1. Schemat pojedynczej jednostki ETLPE
Fig. 1. ETLPE unit framework

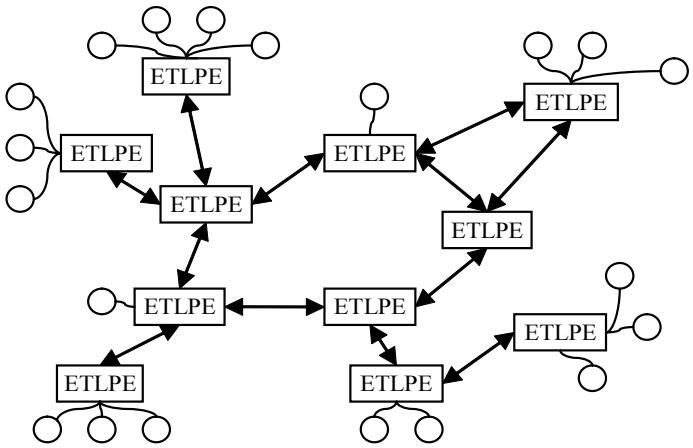
Budowa wewnętrzna jednostki ETLPE oparta jest na trzech podstawowych filarach:

- *Repozytorium zasobów* – gromadzi i udostępnia zarejestrowane zasoby, przeprowadza proces rejestracji, którego celem jest wstępne przetestowanie dostępności zasobu.
- *Warstwa komunikacji* – zapewnia interfejs komunikacji pomiędzy sąsiednimi inżynierami ETL. Dla każdego połączenia tworzony jest osobny komunikator, obsługujący protokół

komunikacji. Warstwa komunikacji odpowiedzialna jest również za zbieranie informacji o parametrach połączenia (wadliwości, przepustowości).

- *Menadżer lokalnych procesów* – obsługuje i dba o poprawne wykonywanie zdefiniowanych lokalnie procesów, udostępnia interfejs komunikacji z lokalnie zarejestrowanym procesem. Każdy proces posiada ponadto menadżera komponentów, którego zadaniem jest tworzenie i usuwanie lokalnych węzłów procesu i obsługa ich pracy.

Warstwa sterowania odpowiada za pełną funkcjonalność ETLPE, na którą może wpływać użytkownik. Zapewnia tym samym bardzo elastyczny interfejs, który można wykorzystać przy integracji z innymi systemami.



Rys. 2. Schemat sieci ETLPE wraz z zarejestrowanymi zasobami
Fig. 2. ETLPE network with registered resources diagram

3.1. Budowa sieci ETLPE

W pierwszej kolejności zostanie omówiony ogólny model ETLPE oraz budowa sieci połączeń (rys 2). Kolejnym zagadnieniem będzie repozytorium zasobów i warstwa komunikacji, a następnie lokalny graf ETL oraz optymalizator projektu.

3.1.1. Identyfikacja obiektów ETLPE

W rozproszonym środowisku przy założeniu niezależności każdego z jego elementów zachodzi potrzeba identyfikacji i rozróżniania obiektów. W sieci ETLPE występują następujące elementy wymagające unikalnej identyfikacji:

- Serwery ETLPE – mogą być tworzone całkowicie poza zdefiniowaną wcześniej siecią i przyłączane do niej w dowolnym czasie.
- Zasoby – ich lokalna identyfikacja nie stanowi problemu, ale zebrane w zbiorze zasobów globalnych będą obarczone błędem powielania nazw (np. dwa pliki o tej samej nazwie i ścieżce dostępu).

- Procesy – definiowane podobnie jak zasoby – lokalnie, ale uruchamiane są na serii rozproszonych serwerów.

Aby uzyskać spójny i unikatowy system identyfikacji, zaproponowaliśmy specyficzną budowę identyfikatora. Głównym identyfikatorem jest identyfikator inżyniera *EngineerID*. Składa się z nazwy inżyniera, lokalizacji oraz unikalnego kodu identyfikacji tworzonego za pomocą o sprzętowych kodów (między innymi numer MAC karty sieciowej) i liczby losowej opartej o aktualnym czasie tworzenia. Tak utworzony identyfikator jest praktycznie unikalny i niepowtarzalny.

Z tak zdefiniowanym identyfikatorem łatwo utworzyć unikalne identyfikatory zasobu *ResourceID* i procesu *ProcessID*, w których skład wchodzi *EngineerID*, na którym są tworzone, oraz lokalnie unikalny identyfikator.

3.1.2. Repozytorium zasobów

Aby każdy z inżynierów posiadał jednakowy równorzędny interfejs dostępu do źródeł oraz odbiorników krotek, proponujemy system oparty na rejestrowaniu lokalnych zasobów i zbieraniu ich reprezentantów na żądanie dowolnej jednostki ETLPE.

Każdy zasób składa się z dwu części:

- Właściwego obiektu – będącego dowolnym źródłem lub odbiornikiem takich danych, które można przedstawić w postaci krotek.
- Reprezentanta – przechowuje wszystkie istotne informacje cechujące dany obiekt i na ich podstawie jest w stanie utworzyć odpowiedni zestaw instrukcji wymaganych dla utworzenia obiektu lokalnego węzła procesu ETL.

Rejestracja zasobu przeprowadzana jest za pomocą pliku konfiguracyjnego, w którym opisane są cechy obiektu oraz jego reprezentant. Przykładowa budowa pliku konfituracyjNego docelowego zasobu w formie pliku tekstowego przedstawiona została na rysunku 3. Do podstawowych cech zasobu zaliczają się parametry: *name* – określający unikalną nazwę zasobu w lokalnym systemie, *type* – definiujący klasę reprezentanta zasobu oraz dedykowany węzeł (w tym przypadku dla parametru *direction* o wartości IN będzie to operator wstawiania) określony parametrem *class*. Nazwy poszczególnych atrybutów zasobu i ich typy definiowane są parametrami *attrNames* i *attrTypes*. Pozostałe parametry są specyficzne dla danego zasobu i ich specyfikacja może być dowolna, odpowiadająca określonemu charakterowi obiektu. Budowa i działanie lokalnego grafu zostanie omówiona w rozdziale 3.1.4.

```
res.name         = TestFile
res.type         = RTextFileInserter
res.class        = TextFileInserter
res.direction    = IN
res.attrNames    = ID,   Name,   SName,  Age,    Money
res.attrTypes    = I,    S,      S,      S,      F
res.path         = d:\\temp\\in.txt
res.isHeader     = true
res.sepRow       = \r\n
res.sepCollumn   = \t
```

Rys. 3. Plik konfiguracyjny zasobu w formie pliku tekstowego
Fig. 3. Resource configuration file

Po wczytaniu konfiguracji następuje utworzenie odpowiedniego reprezentanta i rozpoczyna się proces walidacji, mający na celu potwierdzenie dostępności zasobu. Jeśli powyższa procedura zakończy się poprawnie, reprezentant zasobu zostanie złożony w repozytorium i od tej chwili może być udostępniany dowolnej jednostce inżyniera, wchodzącej w skład sieci ETLPE.

3.1.3. Warstwa komunikacji

Warstwa komunikacji jest obszarem wymiany komunikatów i zadań pomiędzy rozproszonymi jednostkami inżynierów. Każdy z inżynierów może bezpośrednio komunikować się jedynie z sąsiednimi (bezpośrednio połączonymi) serwerami. Komunikacja z pozostałymi jednostkami jest możliwa za pomocą o pośrednictwa. Transmisję komunikatów i zdań można podzielić na trzy typy:

- rozgłaszane – dedykowane wszystkim jednostkom ETLPE (pobieranie globalnie zdefiniowanych zasobów, pobieranie informacji o połączeniach),
- dedykowane – zawierające ścieżkę propagacji zadania do danej jednostki ETLPE,
- bezpośrednie – bezpośrednia komunikacja z sąsiednim serwerem (testowanie przepustowości łącza).

W skład warstwy komunikacji wchodzi serwer komunikacji (odpowiedzialny za przejmowanie i obsługę zgłoszeń zewnętrznych) oraz repozytorium połączeń (przechowuje i obsługuje połączenia z sąsiednimi serwerami).

Łączenie sąsiednich ETLPE, tworzenie sieci ETLPE

ETLPE dla poprawnego działania nie musi być częścią większej struktury, w takim przypadku działa jedynie opierając się na lokalnie zdefiniowanych zasobach. Aby uruchomić ETLPE, należy zdefiniować nazwę i lokację, w której zostanie uruchomiony (nazwa i lokacja są dowolne i zależą od administratora). Na ich podstawie oraz na podstawie sprzętowych parametrów komputera zostanie utworzony identyfikator inżyniera *EnginnerID*. Dodatkowo wymagane jest podanie portu, na którym zostanie uruchomiony serwer komunikacji.

Do tak przygotowanego systemu możemy dołączać kolejne serwery ETLPE. Proces uruchamiania przebiega podobnie, z tym że dodatkowo definiujemy połączenie do już

istniejącego ETLPE. Serwer, z którego nawiązujemy połączenie, wysyła dwa żądania. Pierwsze, zawierające lokalny identyfikator inżyniera, jest prośbą o identyfikator zdalnego inżyniera, a drugie, żądaniem nawiązania połączenia zwrotnego. W rezultacie otrzymujemy parę równorzędnych ETLPE. Proces przyłączania kolejnych jednostek może być kontynuowany bez większych ograniczeń bez względu na fizyczną strukturę sieci. Jedynym wymogiem takiego łączenia jest możliwość utworzenia na obu serwerach gniazda serwera i otwarcie dwóch kanałów komunikacyjnych.

Pomiar przepustowości połączenia

Ogólnie znanym faktem jest to, że czas transmisji dwóch jednakowych pakietów danych nie musi zależeć od geograficznej odległości pomiędzy serwerami. Czas transmisji zależy od medium transmisyjnego, aktualnego obciążenia, protokołu oraz zakłóceń na trasie pakietu. Dla potrzeb projektu sieci ETLPE zdefiniowano parametr, zwany dalej odległością sieciową, rozumiany jako czas dwustronnej transmisji stałego pakietu pomiędzy parą węzłów sieci. Ponieważ odległość sieciowa nie jest stała i zależy od wielu czynników, należy ją badać przez:

- Dokonywanie serii częstych pomiarów (uzależnionych od pory dnia, tygodnia) porównywanych z wynikami z wcześniejszych pomiarów – najdokładniejsza metoda; otwartą kwestią jest, jak często dokonywać pomiarów i jak wiele ich magazynować.
- Wyliczenie średniej z serii pomiarów i średniej przepustowości łącza w pewnym okresie (otrzymany wynik może być zaszumiony chwilową niedostępnością łącza).
- Utrzymywanie tablicy zawierającej n pomiarów, która przy każdorazowej aktualizacji odrzuca pomiar najbardziej odbiegający od średniej – małe obciążenie związane ze składowaniem pomiarów (wystarczy kilkanaście), odporność na zaszumienie wyników, w rezultacie nie otrzymamy aktualnej przepustowości, ale maksymalną.

W naszym przypadku korzystamy z wersji tablicowej (3), ponieważ cechuje się ona stabilnością, jest tym samym dość dobrze dopasowana dla środowiska ze statyczną alokacją węzłów.

Po nawiązaniu połączenia pomiędzy parą sąsiadujących inżynierów rozpoczyna się periodyczny proces pomiaru parametrów transmisji. Okres tych pomiarów jest konfigurowalny i zależny od charakteru danego środowiska. Pomiar polega na utworzeniu pakietu transmisyjnego o zdefiniowanym dla wszystkich inżynierów jednakowym rozmiarze, rozpoczęciu pomiaru czasu, przesłaniu go do sąsiedniego inżyniera i oczekiwaniu na jego powrót. Następnie jednostka dokonująca pomiaru oblicza jego wynik i zwraca go sąsiedniej w celu zaoszczędzenia zbędnego obciążania łącza.

Zbieranie rozproszonych zasobów i pobieranie informacji o strukturze połączeń

Każdy ETLPE możne zarejestrować dowolne zasoby (pkt 3.1.2). Tak zdefiniowane zasoby są widoczne dla wszystkich inżynierów wchodzących w skład sieci ETLPE. Aby

zebrać wszystkie globalnie zdefiniowane zasoby, inżynier odpytuje wszystkie węzły sieci w następujący sposób:

- tworzone jest zapytanie złożone z wektora identyfikatorów zawierającego identyfikatory odwiedzonych lokacji (wstępnie zainicjowany jest lokalnym identyfikatorem),
- zapytanie przesłane jest kolejno do wszystkich sąsiednich inżynierów, którzy nie figurują w wektorze odwiedzonych serwerów,
- po odebraniu zapytania sąsiedni inżynier dodaje swój identyfikator do wektora odwiedzonych lokacji i propaguje zapytanie do wszystkich inżynierów, których identyfikator nie znalazł się jeszcze w wektorze,
- w odpowiedzi na zapytanie każdy z serwerów zwraca sumę zbioru otrzymanego jako odpowiedź zapytania i zbioru lokalnych zasobów w formie ich reprezentantów.

Dzięki zastosowaniu propagowanych wektorów z identyfikatorami odwiedzonych lokalizacji zapobiegamy występowaniu pętli cyklicznych. Wynikiem powyższej procedury jest zbiór globalnie zdefiniowanych zasobów w formie ich reprezentantów.

W podobny sposób przebiega proces odpytywania systemu o strukturę połączeń. Każdy z ETLPE posiada informacje o sąsiednich połączeniach i parametrach tego połączenia, więc zapytanie oparte na powyższych zasadach zwróci wynik, będący kolekcją informacji w formie {*EnginnerIDA*, *EnginnerIDB*, *dstAB*}, gdzie *EnginnerID* to identyfikator inżyniera, a parametr *dst* określa dystans sieciowy pomiędzy nimi.

W odniesieniu do powyższych rozważań należy nadmienić, że w przypadku znacznie rozproszonych systemów zaproponowany wyżej algorytm może okazać się mało wydajny i wrażliwy na zakłócenia. Alternatywą dla tego przypadku jest rozgłaszanie reprezentantów zasobów po całej sieci w momencie ich rejestracji, a także rozgłaszanie komunikatu o ich wyrejestrowaniu oraz utrzymywanie aktualnego ich stanu na każdym z inżynierów procesu. Rozwiązanie to pozwoliłoby na łatwe potwierdzenie lokalnego stanu opierając się na stanie sąsiada. Wprowadzane obciążenie sieciowe byłoby jednostkowe i występowałoby jedynie w czasie rozgłoszenia zasobu lub komunikatu o wyrejestrowaniu.

Rozsyłanie zadań i komunikacja w procesie

Zadania dla procesu ETL (proces tworzenia zadań oraz ich budowa zostaną szczegółowo omówiona w rozdziale 3.1.5) zawierają wektor propagacji, definiujący drogę (w postaci kolejnych identyfikatorów *EnginnerID* w kierunku od docelowego do źródłowego inżyniera) transmisji zadania. Na podstawie takiej drogi i indeksu pozycji (określającego aktualne miejsce na drodze) można pobrać kolejny identyfikator inżyniera (sąsiada). W ten sposób ETLPE mogą przekazywać sobie zadania niezależnie od struktury warstwy połączeń sieciowych.

Automatyczna konfiguracja połączeń pomiędzy rozproszonymi lokalnymi procesami

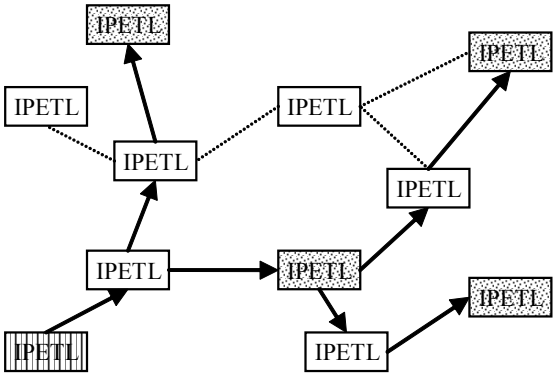
Po rozesłaniu zadań do każdego z uczestniczących w procesie inżynierów tworzony jest lokalny graf ekstrakcji (punkt 3.1.4). Po ich utworzeniu następuje proces łączenia poszczególnych węzłów procesu (lokalnych grafów). Obecnie transmisja krotek pomiędzy węzłami procesu odbywa się na podstawie protokołu TCP/IP.

Optymalizator procesu nie może sztywno określać fizycznych parametrów połączeń, takich jak port nasłuchu dla operatora zdalnego wstawiania. W przypadku zajętości zasobu wystąpi błąd tworzenia operatora i tym samym blokada procesu. Ponadto, wielu niezależnych użytkowników posługujących się różnymi serwerami ETLPE, tworząc procesy ETL korzystające z danej jednostki, wykorzystuje jej zasoby w sposób niezależny. Aby rozwiązać ten problem, zaproponowano następujący mechanizm:

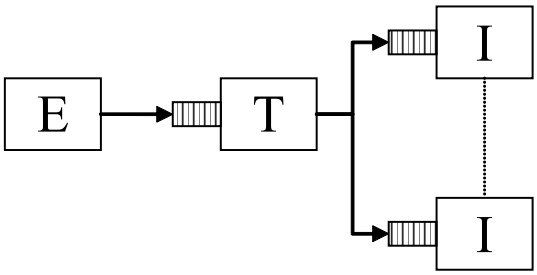
- każdy serwer lokalnie definiuje zakres dostępnych portów,
- operatory zdalnej komunikacji otrzymują wspólny identyfikator autokonfiguracji,
- podczas tworzenia operatora zdalnego wstawiania na jednym wolnym porcie (należącym do zdefiniowanego zakresu) tworzone jest gniazdo serwera i wraz z identyfikatorem autokonfiguracji składowane jest w lokalnym repozytorium gniazd danego procesu i przydzielane operatorowi zdalnego wstawiania,
- podczas tworzenia operatora zdalnej ekstrakcji menadżer komponentów za pośrednictwem warstwy komunikacji odpytuje sąsiednich inżynierów o parametry gniazda, identyfikowane parametrem autokonfiguracji,
- na podstawie pobranych informacji operator zdalnej ekstrakcji tworzy gniazdo klienta i nawiązuje łączność.

3.1.4. Lokalny i globalny graf ETL

Koncepcja rozproszonego strumieniowego procesu ETL oparta jest na lokalnych grafach ekstrakcji, które powstają na odpowiednio dobranych jednostkach ETLPE i grafie globalnym, utworzonym w wyniku połączenia grafów lokalnych (rys. 4, 5). Schemat połączeń powstaje na bazie dróg sieciowych. Graf lokalny zbudowany jest z węzłów ekstrakcji oraz krawędzi pełniących rolę pośrednich buforów pomiędzy węzłami. Budowany jest automatycznie przez Menadżera Komponentów na podstawie skryptu konfiguracji.



Rys. 4. Globalny graf ETL. Strzałki wskazują drogi przepływu strumienia od ETLPE zasilającego do docelowych
Fig. 4. Global ETL graph diagram. Arrows points stream flow from source to receivers



Rys. 5. Schemat lokalnego grafu ETL
Fig. 5. Local ETL graph diagram

Węzły procesu ETL można podzielić na trzy grupy:

- Węzły ekstrakcji – węzły odpowiedzialne za pobieranie danych ze źródła oraz ich konwersję na format krotek. W tej grupie występują ekstraktory, które powstają na podstawie instrukcji reprezentanta zasobu lub zdalnych ekstraktorów łączących lokalne grafy w graf globalny.
- Węzeł transformacji – wszelkiego typu operacje na krotkach. Szczególnym operatorem w naszym systemie jest *Spliter* (podzielnik strumienia).
- Węzły wstawiania – węzły odpowiedzialne za wstawianie danych do odpowiednich odbiorników. W tej grupie występują operatory wstawiania powstałe na podstawie instrukcji reprezentanta zasobu lub zdalnych operatorów wstawiania łączących lokalne grafy w graf globalny.

Krawędzie (zwane też kanałami) pomiędzy węzłami są buforami służącymi do transmisji krotek pomiędzy elementami przetwarzania ETL. Mogą być implementowane i konfigurowane w dowolny sposób. Na potrzeby naszego projektu przyjęliśmy bufory w formie kolejek pamięciowo-dyskowych.

Każdy z operatorów działa niezależnie od innych opierając się na krotkach znajdujących się w swoim buforze. Przetwarzanie każdego z węzłów składa się z pobrania krotki danych,

przetworzenia jej w zależności od charakteru operatora i przesłania jej kolejnemu elementowi procesu. W przypadku braku krotek w buforze operator przechodzi w stan uśpienia i pozostaje w nim, dopóki nie wystąpi jedno z dwu zdarzeń:

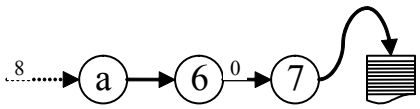
- operator zasilający bufor wstawi do niego krotkę danych i wywoła zdarzenie budzące,
- upłynie zdefiniowany interwał czasowy.

W przypadku zapełnienia 80% bufora pamięciowego operator wstawiający po każdorazowym przetworzeniu krotki oddaje swój czas procesora i tym samym spowalnia swoje przetwarzanie.

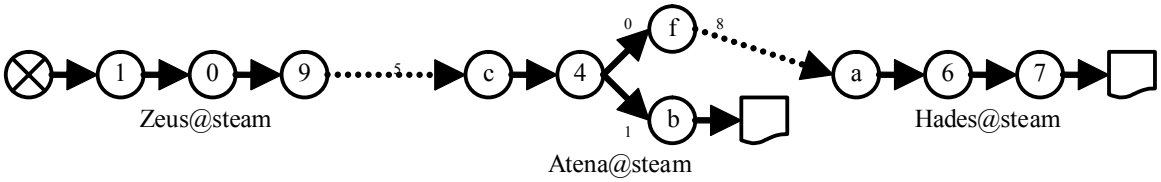
Jak wspomniano wcześniej, lokalne grafy tworzone są na podstawie odpowiedniego, dedykowanego dla danego ETLPE skryptu konfiguracyjnego, który powstaje w procesie projektowania zadań.

```
6.out.0              = 7
6.out.0.size         = 512
6.out.0.type         = MDEdge
6.type               = Spliter
7.atrNames           = ID, Name, SName, Age, Money
7.atrTypes           = INTEGER, STRING, STRING, STRING, FLOAT
7.fieldsSeparator    = ;
7.fileName           = e:\getl\temp\in.txt
7.isHeader           = false
7.linesSeparator     = \n
7.type               = TextFileInserter
a.out                = 6
a.out.size           = 512
a.out.type           = MDEdge
a.remAutoConfigId    = 8
a.timeout            = 1000
a.type               = RemoteExt
system.nodes         = 6,7,a
```

Rys. 6a. Skrypt konfiguracyjny
Fig. 6a. Configuration script



Rys. 6b. Lokalny graf ETL
Fig. 6b. Local ETL graph



Rys. 6c. Pełny graf ekstrakcji
Fig. 6c. Full ETL graph

Budowa instrukcji w skrypcie oparta jest na schemacie

identyfikator.parametr[.parametr]=wartość

gdzie: *identyfikator* określa unikalny w procesie identyfikator węzła lub parametr systemowy system, *parametr* jest cechą identyfikatora. Menadżer komponentów w pierwszej kolejności interpretuje parametry systemowe, a przede wszystkim odczytuje listę zdefiniowanych lokalnie węzłów *system.nodes* i dla każdego z nich tworzy obiekt określony parametrem

id_węzła.type (rys. 6a i 6b). Na rys. 6c zaprezentowany został pełny graf ekstrakcji, zbudowany z trzech skryptów konfiguracji, po jednym dla każdego serwera.

3.1.5. Optymalizator projektu sieci ETLPE

Pierwszym etapem po zebraniu informacji o zasobach jest wyznaczenie pośrednich, docelowych i źródłowych inżynierów procesu ETL. Aby uzyskać tę informację, algorytm pobiera następujące parametry wejściowe:

- kolekcję złożoną z niepowtarzalnych obiektów o atrybutach $EngineerID_A$, $EngineerID_B$, odległość sieciowa pomiędzy *A*-*B*,
- tablicę zdefiniowanych zasobów z wyróżnionym indeksem zasobu zasilającego,
- identyfikator lokalnego inżyniera procesu ETL.

Jako wyniku działania algorytmu spodziewamy się kolekcji zadań dla każdego z wyznaczonych inżynierów procesu. Kolekcja zadań składa się z następujących parametrów:

- identyfikator procesu,
- plik konfiguracyjny zawierający skrypt instrukcji tworzenia lokalnego grafu ETL,
- tablicę kolejnych identyfikatorów inżynierów określającą kierunek propagacji zadania pomiędzy inżynierami w kierunku od docelowego do lokalnego serwera.

Ogólna struktura algorytmu jest następująca:

- pobierz kolekcję informacji o połączeniach,
- wyznacz tablicę zawierającą tablicę najkrótszych dróg (kolejnych identyfikatorów inżynierów) – w naszym przypadku korzystamy z algorytmu Dijkstry [5],
- generuj kolejne pliki konfiguracyjne dla każdego z pośredniczących inżynierów,
- wyznacz drogi propagacji zadań.

Generowanie plików konfiguracyjnych

Algorytm wykonuje się dla zbioru zdefiniowanych reprezentantów zasobów. Z puli zdefiniowanych zasobów wybierany jest dowolny zasób i z wektora dróg pobierana jest odpowiadająca mu droga (identyfikator zasobu jest zgodny z pierwszym identyfikatorem inżyniera tablicy drogi).

Kolejnym krokiem jest iteracja po wszystkich inżynierach zdefiniowanych w drodze i tworzenie pośrednich podzadań. Podzadania przechowywane są w tablicy mieszającej i identyfikowane na podstawie docelowego identyfikatora inżyniera. Jeśli dla danego identyfikatora inżyniera tablica mieszająca zwróci wartość pustą, oznacza to, że nie został jeszcze dla niego zdefiniowany obiekt zadania. Parametrami zadania są: docelowy identyfikator inżyniera, identyfikator procesu oraz konfiguracji zapisanej w skrypcie konfiguracyjnym. Plik taki w wersji początkowej składa się jedynie z jednego operatora – *Splitera* (podzielnika strumienia) bez zdefiniowanych połączeń.

Kolejnym krokiem jest podpięcie danego zasobu do konfiguracji pliku. Dokonywane jest to w następujący sposób:

- każdy zasób na podstawie nazwy (identyfikatora) operatora potrafi zwrócić poprawną konfigurację dla procesu. Na jej podstawie tworzony jest operator lokalnego grafu ETL,
- jeśli zasób jest producentem (zasób zasilający), operatorem jest ekstraktor i definiuje się dla niego podłączenie do głównego operatora *Splitera*,
- jeśli zasób jest konsumentem, jego operatorem jest operator wstawiania i połączenie jest definiowane ze *Spliterem*.

Następnie wykonane są połączenia pomiędzy rozproszonymi inżynierami. Zauważyć należy, że kolejność propagacji zadań na ścieżce jest zdefiniowana w kierunku przeciwnym do przepływu krotek. Połączenie dokonywane jest, gdy tablica drogi posiada więcej niż jeden element (można wyznaczyć element wcześniejszy i-1). Tak więc aktualny i-ty element jest dla nas źródłem krotek, a i-1 ich odbiorcą. Dla elementu i-tego tworzony jest operator zdalnego wstawiania krotek, a i-1 element wzbogacany jest o operator zdalnej ekstrakcji. Oba operatory otrzymują unikalny identyfikator *remAutoConfigID* (w obrębie procesu) w celu automatycznej konfiguracji (za pomocą mechanizmu komunikacji pomiędzy inżynierami). Ekstraktor tworzy połączenie do lokalnego operatora podzielnika strumienia, a operator ładowania jest podłączany do jego lokalnego podzielnika.

Gdy i-te zdanie posiada plik konfiguracyjny ze zdefiniowanym już operatorem zdalnej ekstrakcji, oznacza to, że uczestniczy już w procesie zasilania innego zasobu i proces tworzenia połączeń pomiędzy inżynierami może zostać zakończony.

W ten sposób uzyskiwany jest efektywny graf ekstrakcji.

Wyznaczanie drogi propagacji zadań

Dokonywane jest w podobny sposób jak wyznaczanie tablicy najkrótszych dróg z tą różnicą, że jako parametr startowy algorytmu Dijkstry podawany jest indeks lokalnego identyfikatora inżyniera.

3.1.6. System rozproszonej kontroli procesu i detekcja błędów

Nawet w przypadku zastosowania najbardziej wydajnego sprzętu i najlepszego oprogramowania nie możemy zakładać braku awaryjności systemów. Ponadto, szansa na pojawienie się awarii jest tym większa, im bardziej system jest rozbudowany, a co za tym idzie, zwiększa się jego skala złożoności. Możemy spodziewać się następujących typów błędnego funkcjonowania systemu: zatrzymanie lokalnego procesu, zatrzymanie lub awaria jednostki ETLPE, błędne działanie dowolnego elementu systemu, powodujące w efekcie zatrzymanie strumienia krotek lub jego zakłócenie.

Zamierzamy wprowadzić protokół *heartbeat* do detekcji błędów związanych z długotrwałym rozłączeniem lub rekonfiguracją sieci. Błędy transmisji mogą być wykrywane przez

zaimplementowany system periodycznego testowania przepustowości sieci. Najtrudniejsze w wykryciu są błędy związane z niepoprawnym przetwarzaniem krotek, gdzie pomimo niezgłaszania komunikatów o wystąpieniu pewnych anomalii proces nieprzerwanie przetwarza krotki i zwraca błędne wyniki. Kwestia obsługi trzeciej kategorii błędów jest przedmiotem kolejnych badań związanych z rozszerzeniem funkcjonalności menadżera procesu.

W danej chwili system rozproszonej kontroli procesu pozwala na zbieranie aktualnego stanu poszczególnych operatorów. Komunikacja z procesem oparta jest na transmisji dedykowanej – z dowolnego serwera można wysyłać dedykowane zadania dla pojedynczego węzła procesu, a także dla całego jego zbioru. W chwili obecnej zdefiniowane są trzy podstawowe zadania:

- zarejestrowanie zadań,
- uruchomienie procesu,
- zatrzymanie procesu,
- zebranie dokładnych informacji o stanie lokalnych operatorów i menadżera komponentów.

Podglądanie logów z lokalnie pracujących procesów

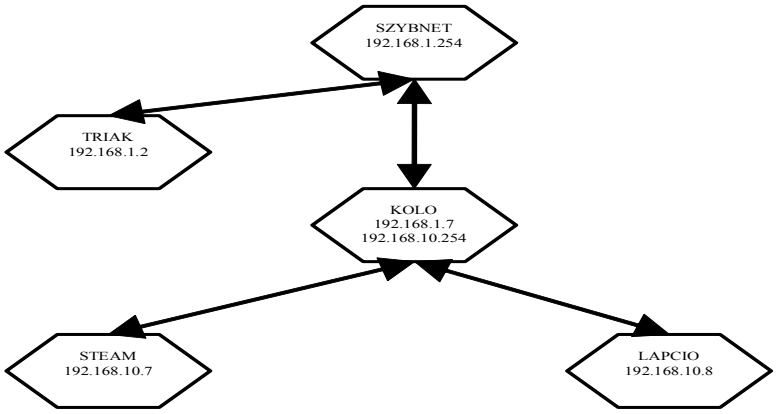
W chwili rejestracji zadań na każdym z ETLPE, należącym do danego procesu, tworzony jest wirtualny zasób udostępniający, w formie logów przekształconych do postaci krotek, informacje o aktualnym stanie przetwarzania. Tym samym można utworzyć osobny proces zbierający powyższe informacje i rejestrujący je w odpowiednio zdefiniowanym odbiorniku krotek.

4. Testy funkcjonalne

Zparezentowane w tym rozdziale testy mają na celu przedstawienie i sprawdzenie możliwości sieci ETLPE. Wykorzystana zostanie sieć LAN o budowie hierarchicznej pokazano na rys. 7.

Parametry komputerów:

- TRIAK – procesor Athlon XP 2GHz, pamięć operacyjna 1GB, dysk 160GB,
- KOLO – procesor Sempron 1,6GHz pamięć operacyjna 512MB, dysk 70GB, pełni funkcje routera,
- STEAM – procesor Athlon 64 DualCore 2GHz, pamięć operacyjna 2GB, dysk 2x160GB (RAID-0),
- LAPCIO – procesor Athlon XP 1,8GHz, pamięć operacyjna 512MB, dysk 40GB,
- SZYBNET – router.



Rys. 7. Środowisko testowe – lokalna sieć LAN
Fig. 7. Test environment – LAN

Format docelowego pliku A:

```
res.name        = AFormatFile
res.type        = RTextFileInserter
res.class       = TextFileInserter
res.direction   = IN
res.attrNames   = ID,   Name,   SName,  Age,    Money
res.attrTypes   = I,    S,      S,      S,      F
res.path        = d:\\temp\\in.txt
res.isHeader    = true
res.sepRow      = \r\n
res.sepCollumn  = \t
Format docelowego pliku B:
res.name        = BFormatFile
res.type        = RTextFileInserter
res.class       = TextFileInserter
res.direction   = IN
res.attrNames   = ID,   Name,   SName,  Age,    Money
res.attrTypes   = I,    S,      S,      S,      F
res.path        = d:\\temp\\inB.txt
res.isHeader    = true
res.sepRow      = %
res.sepCollumn  = ,
```

4.1. Lokalny proces na pojedynczym komputerze

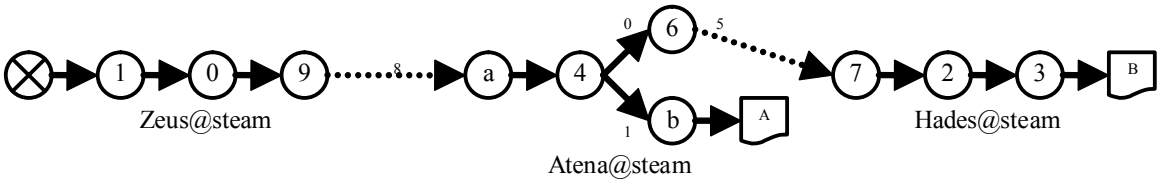
Celem tego testu jest wykazanie możliwości lokalnej pracy serwera oraz utworzenia wielu niezależnych serwerów na jednej jednostce obliczeniowej. Dodatkowo zostaną zmniejszone pojemności buforów (krawędzi) w celu sprawdzenia działania mechanizmów kolejki pamięciowo-dyskowej oraz wygenerowany zostanie strumień krotek o dużym natężeniu. Parametry testowe są następujące: źródło danych – generator krotek, dwa odbiorniki danych – pliki tekstowe o różnych parametrach formatowania, liczba krotek – 100000, pojemność pamięciowego bufora kanału – 512 elementów, liczba komputerów – 1 (STEAM), liczba serwerów ETLPE – 3.

Na komputerze STEAM uruchamiamy trzy niezależne serwery ETLPE: Zeus, Atena i Hades. Ich warstwy komunikacyjne działają na portach 1982, 1983, 1984. Serwery te otrzymują następujące identyfikatory: Zeus@steam, Atena@steam i Hades@steam. Z serwera Atena nawiązujemy połączenia do dwu pozostałych serwerów, w ten sposób tworzymy grupę szeregowo powiązanych serwerów ETLPE. Na serwerze Zeus rejestrujemy zasób generatora 100 000 krotek, generowanych bez opóźnienia, na Atenie i Hadesie zasoby docelowe – pliki tekstowe w dwu różnych formatach bazujących na CSV. Tworzymy projekt na serwerze Hades, ustawiając jako zasób zasilający generator krotek, a zasoby do zasilenia – pliki tekstowe.

Ponieważ wygenerowany plik konfiguracyjny jest za długi, by mógł być częścią artykułu, prezentujemy tutaj jego wycinek i graficzną reprezentację (rys.8) całego grafu.

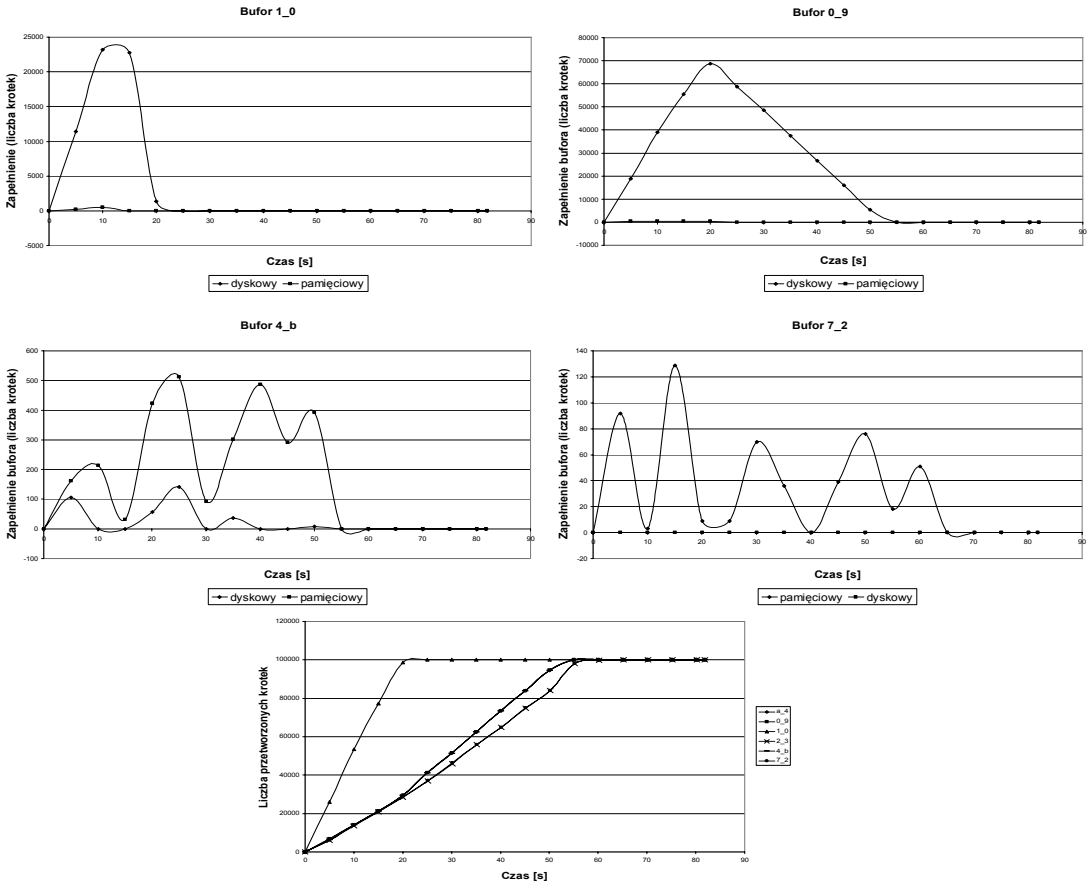
```
Lokacja: Atena@steam
ID procesu: Hades@steam:0
Plik konfiguracyjny:
4.out.0   = 6
4.out.0.size  = 512
4.out.0.type  = MDEdge
4.out.1   = b
4.out.1.size  = 512
4.out.1.type  = MDEdge
4.type = Spliter
6.remAutoConfigId   = 5
6.timeout = 1000
6.type = RemoteIns
a.out  = 4
a.out.size= 512
a.out.type= MDEdge
a.remAutoConfigId   = 8
a.timeout = 1000
a.type = RemoteExt
b.atrNames= ID,Name,SName,Age,Money
b.atrTypes= INTEGER,STRING,STRING,STRING,FLOAT
b.fieldsSeparator   =
b.fileName= d:\temp\inA.txt
b.isHeader= true
b.linesSeparator =

b.type = TextFileInserter
id  = Hades@steam0
system.nodes  = 4,6,a,b
Ścieżka propagacji zadań:
Hades@steam -> Atena@steam
```

Powyższy plik konfiguracyjny przekłada się na poniższy schemat



Rys. 8. Graficzna reprezentacja wygenerowanego grafu
Fig. 8. Graphical representation of configuration file

Rozsyłamy zadania na każdym z uczestniczących w procesie serwerów ETLPE, sprawdzamy, czy w panelu projekt pojawił się odpowiedni wpis i możliwość przejścia do okna logów procesu. Następnie uruchamiamy proces ETLPE i obserwujemy wpisy do logów dla każdego z podprocesów.



Rys .9. Wykres zapełnienia buforów
Fig. 9. Buffer fill graph

Jak można zauważyć na rys. 9, gdy liczba krotek w buforze pamięciowym przekroczy maksymalny pułap 512 elementów, kolejne krotki są zapisywane do bufora dyskowego. W przypadku gdy bufor pamięciowy się opróżni, następuje pobieranie krotek z bufora dyskowego. Zapełnianie się bufora dyskowego spowodowane jest niezdolnością węzła odbiorczego do pobierania informacji z bufora i nadproduktywnością węzła nadawczego. Powyższe wykresy zostały zaobserwowane dla kanałów 1_0, 0_9, 4_b, 7_2 i są wynikiem opóźnień powodowanych przez dostęp do zasobów dyskowych i na komputerze lokalnym.

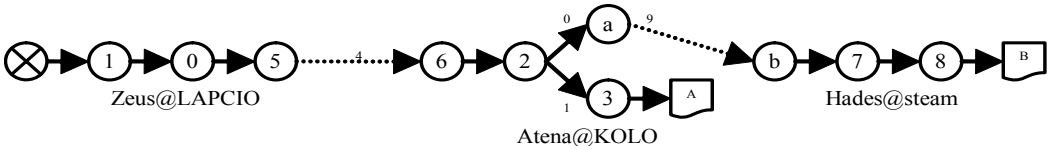
Zatem, generator krotek nie jest zależny od zasobów systemowych i generuje krotki z prędkością, na jaką pozwala moc obliczeniowa komputera STEAM (co prowadzi do przepełnienia bufora pamięciowego 1_0 i 0_9). Ponieważ operator 9 przesyła krotki z pewnym opóźnieniem spowodowanym dostępem do zasobu dyskowego, pozostałe operatory mają wystarczające okna czasowe, by przetworzyć krotkę. Różnica przetwarzania pomiędzy

kanałami 1_0 oraz 0_9 i 4_b a pozostałymi spowodowana jest dodatkowymi czasami zapisu i odczytu z pamięci dyskowej.

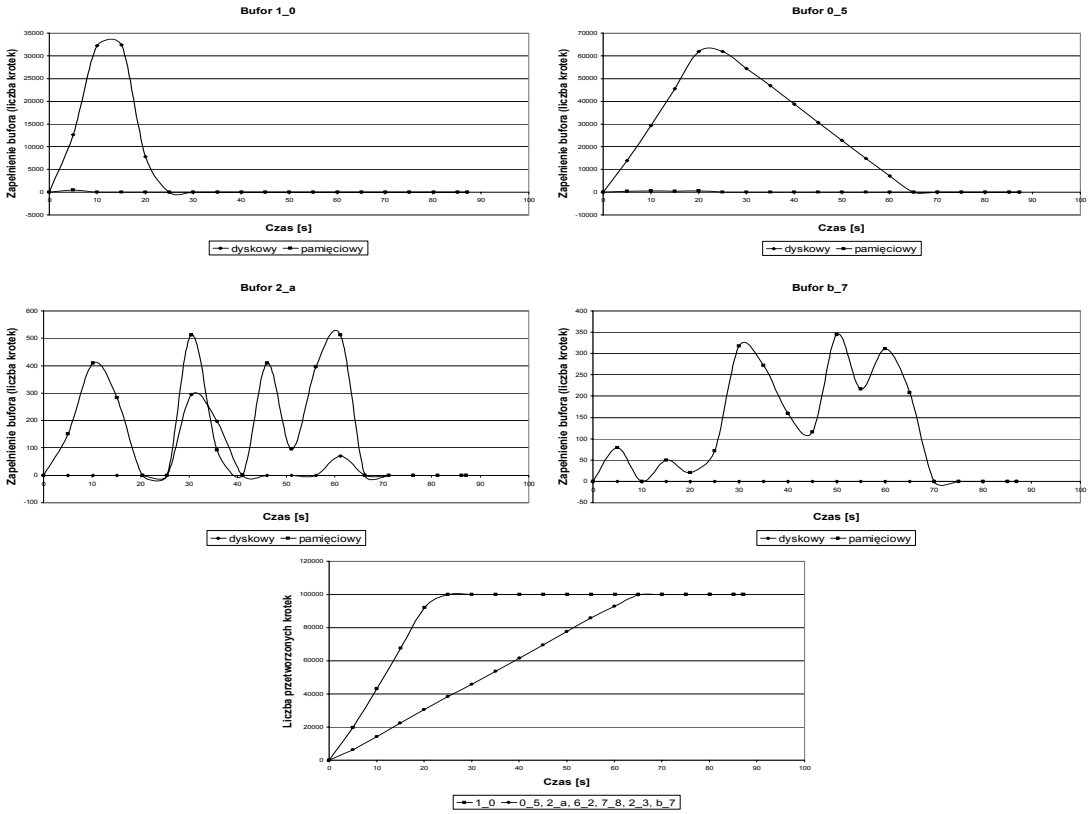
4.2. Proces na równorzędnej sieci LAN

Celem tego testu jest sprawdzenie funkcjonalności warstwy komunikacyjnej i menadżera procesu w środowisku rozproszonym. Testowano: nawiązywanie połączeń, zbieranie informacji o rozproszonych zasobach, działanie procesu w środowisku rozproszonym. Dodatkowo, zostały obniżone pojemności buforów (krawędzi), aby sprawdzić działanie mechanizmów kolejki pamięciowo-dyskowej. W tym celu wygenerowany zostanie strumień krotek o dużym natężeniu. Przyjmujemy następujące parametry testowe: źródło danych – generator krotek, dwa odbiorniki danych – pliki tekstowe o różnych parametrach formatowania, liczba krotek – 100 000, pojemność pamięciowego bufora kanału – 512 elementów, liczba komputerów – 3 (STEAM, LAPCIO, KOLO), liczba serwerów ETLPE – 3 (Zeus, Hades, Atena). Na komputerze LAPCIO uruchamiamy serwer Zeus, na komputerze STEAM serwer Hades i na komputerze KOLO serwer Atena. Serwery te otrzymują następujące identyfikatory: Zeus@LAPCIO, Hades@STEAM i Atena@KOLO.

W tym przypadku nawiązywanie połączeń wykonamy w przeciwnym kierunku, a mianowicie od serwerów Zeus i Hades do Ateny. Na każdym z serwerów rejestrujemy tę samą grupę zasobów w różnych konfiguracjach. Następnie dokonujemy odpytania o wszystkie zasoby. Korzystając z serwera Hades, projektujemy proces o schemacie podobnym do tego z pierwszego testu. Wyniki działania generatora zaprezentowane są na rys. 10.



Rys. 10. Graficzna reprezentacja wygenerowanego grafu
Fig. 10. Graphical representation of configuration file



Rys. 11. Wykres zapełnienia buforów
Fig. 11. Buffer fill graph

Na rys. 11 wyraźnie widać „wąskie gardło” sieci ETLPE. Bardzo duże opóźnienie spowodowane jest przez bufor dyskowy zapchany na kanale 0_5. W takim przypadku następuje przepełnienie bufora pamięciowego i kolejne napływające krotki są składowane w buforze dyskowym. Dodatkowo, dochodzi tutaj problem podziału czasu procesora, który jest jednocześnie odpowiedzialny za zapis i odczyt z bufora dyskowego oraz zapis krotek do interfejsu sieciowego. Opóźnienie spowodowane odczytami i zapisami na dysk powoduje ustabilizowanie transmisji w kolejnych węzłach. Prawdopodobnie dobrym rozwiązaniem byłoby rozpatrzenie optymalizacji kanału pamięciowo-dyskowego o możliwość zapisu i odczytu blokowego. Nie występuje zjawisko blokowania wyjściowych kanałów operatora podzielnika strumienia, dzięki czemu niezależnie od szybkości zasobów ujściowych operator podzielnika strumienia może pracować bez przestojów. Jak możemy zauważyć, serwery ETLPE nie rozróżniają pracy na środowisku lokalnym od pracy na środowisku rozproszonym. Wyniki generowania rozkładu procesu są identyczne.

4.3. Proces na hierarchicznej sieci LAN

Celem tego testu jest sprawdzenie funkcjonalności warstwy komunikacyjnej i menadżera procesu w środowisku rozproszonym oraz wykazanie równorzędności serwerów niezależnie

od topologii sieciowej. Testowane zostanie: nawiązywanie połączeń, wyznaczanie drogi sieciowej, zbieranie informacji o rozproszonych zasobach, wyznaczanie grafu procesu w zależności od drogi sieciowej, działanie procesu w środowisku rozproszonym, ze szczególnym uwzględnieniem hierarchii topologii sieciowej. W celu zbadania powyższych założeń zastosowane zostały następujące parametry testowe: źródło danych – generator krotek, dwa odbiorniki danych – pliki tekstowe o różnych parametrach formatowania, liczba krotek – 600, opóźnienie generowania 500 ms, pojemność pamięciowego bufora kanału – 512 elementów, liczba komputerów – 4 (STEAM, LAPCIO, KOLO, TRIAK), liczba serwerów ETLPE – 4 (Zeus, Atena, Mars, Hades). Na komputerze STEAM uruchamiamy serwer Hades, na LAPCIO Mars, na KOLO serwer Atena i na TRIAK serwer Zeus. Serwery te otrzymują następujące identyfikatory: Hades@STEAM, Mars@LAPCIO, Atena@KOLO i Zeus@-TRIAK (tab. 1).

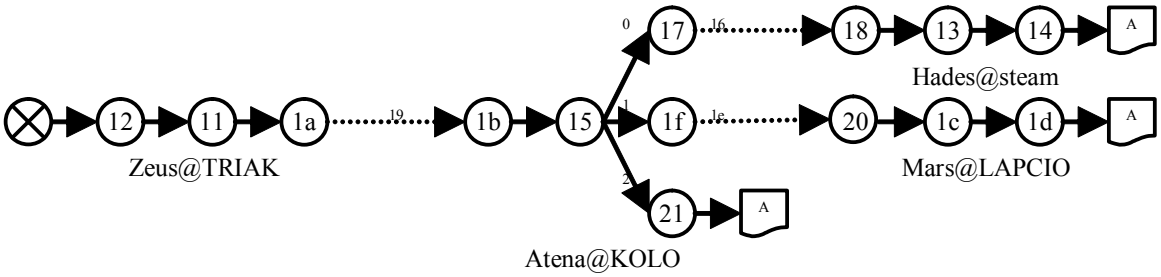
Tabela 1

Pomiary odległości sieciowych

	Atena	Zeus	Hades	Mars
Atena	0	64	46	47
Zeus	59	0	–	–
Hades	48	–	0	50
Mars	45	–	54	0

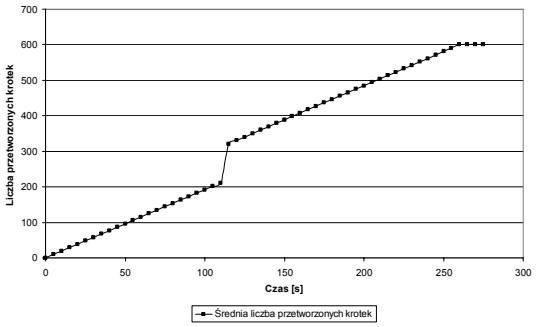
Powyższe wyniki są wyznaczone poprzez serię pomiarów w różnych okresach czasu na nieobciążonych komputerach w lokalnej sieci LAN. Jak możemy zauważyć, nie różnią się one od siebie w znacznym stopniu, jednak odzwierciedlają fizyczny rozkład jednostek obliczeniowych. Komputery, na których stoją serwery Atena, Mars i Hades, znajdują się w jednym pomieszczeniu, komputer TRIAK z serwerem Zeus znajduje się w innym budynku.

Na serwerze Zeus zarejestrowany został zasób generatora, a na Hades, Atena i Mars zasoby docelowych plików tekstowych. Na serwerze Hades został zaprojektowany proces zasilania plików tekstowych znajdujących się na Mars i Hades i Atena z serwera Zeus. Mars i Hades nie mają bezpośredniego połączenia z serwerem Zeus, tak więc w procesie uczestniczyć będzie serwer Atena w formie pośrednika. Ze względu na czytelność wyniki pracy algorytmu definiowania drogi sieciowej będą prezentowane w postaci graficznej



Rys. 12. Graficzny schemat grafu ekstrakcji
Fig. 12. Graphical representation of ETL graph

Projektant powyższego grafu, zgodnie z założeniami projektowania sieci ETLPE, otrzymał jedynie informację o zasobach. Jego rola polegała jedynie na wybraniu zasobu zasilającego i zasobów zasilanych. Jak możemy zauważyć na rys. 12, optymalizator projektu utworzył graf z pośrednictwem na Atena@KOLO, i tym samym zadbał o fizyczną zgodność i logikę transmisji.



Rys. 13. Liczba przetworzonych krotek w funkcji czasu
Fig. 13. Count of processed tuples in time function

Anomalia wykresu w obszarze 125s związana jest z błędnym obliczaniem czasu uśpienia przez generator krotek. Prawdopodobnie w środowisku komputera TRIAK występuje jakiś proces stale aktualizujący czas. Pomiary w środowisku STEAM nie wykazały powyższej anomalii. Błąd ten jednak nie koliduje z przeprowadzanymi testami i nie zakłóca charakteru wykresu. Test ten potwierdza nasze założenia, dotyczące niezależności środowiska pracy od topologii sieciowej. Serwery ETLPE poprawnie wyznaczyły odległości sieciowe, zarejestrowały, udostępniły i zebrały reprezentantów. Serwer Hades poprawnie zdefiniował proces, który został rozesłany do odpowiednich serwerów i poprawnie przez nie przetworzony.

5. Wnioski i uwagi

Artykuł ten nawiązuje tematycznie do systemów przetwarzania strumieniowego na dużą skalę. Zaprezentowano nową koncepcję alternatywnego systemu przetwarzania strumieniowego o równorzędnych węzłach przetwarzania o nazwie sieci ETLPE.

W chwili obecnej rozpatrujemy funkcjonalność sieci ETLPE jedynie pod względem zasilania dużej liczby rozproszonych odbiorników. W najbliższej przyszłości zamierzamy zaimplementować kolejne operatory transformacji, takie jak na przykład złączenie (oparte na oknach, by ograniczyć blokowanie), filtrowanie, projekcję i inne operatory nieblokujące.

W przypadku operatora złączenia algorytm będzie musiał rozpatrywać jego lokalizację w odwrotny sposób do operatora podzielnika strumienia i traktować wynikowe krotki jako źródło danych. Inną koncepcją na implementację operatora złączenia jest jego konfiguracja w formie zasobu wejściowo-wyjściowego. Zasób byłby doładowywany w klasyczny sposób,

który został opisany w poprzednich rozdziałach i udostępniałby krotki, będące wynikiem złączenia strumieni.

Jednym z ważnych aspektów tego systemu jest zarządzanie zasobami. W chwili obecnej system zakłada stałość i niezmienność istnienia i konfiguracji danego zasobu uczestniczącego w aktywnym procesie. Jak możemy zauważyć, w rzeczywistości wskutek różnych nieprzewidzianych okoliczności konfiguracja i istnienie danego zasobu mogą ulec zmianie.

Kolejnym ważnym elementem tematyki rozproszonych zasobów jest ich współdzielenie. W chwili obecnej system zakłada, że z jednego zasobu korzysta tylko jeden proces. Z przeprowadzonych testów wynika, że w kolejnych pracach należy rozpatrzeć utworzenie menadżera zarządzania zasobami oraz zaprojektować system współdzielenia zasobów.

Elementem systemu, na którego zapotrzebowanie wyniknęło w trakcie przeprowadzania testów funkcjonalnych, jest menadżer rozproszonego raportowania statusu procesu i system przejmowania kontroli nad procesem z dowolnego ETLPE, będącego w sieci ETLPE. W chwili obecnej raportowanie o stanie procesu jest znacznie uproszczone i oparte jest jedynie na zbieraniu szczegółowych informacji odnośnie do stanu wszystkich dynamicznych elementów danego podprocesu, oraz lokalnym systemie dziennika. W przyszłych pracach należałoby rozpatrzeć rozszerzenie funkcjonalności raportowania i asynchronicznego zgłaszania problemów. Można by zaproponować rozszerzenie menadżera procesu o lokalnie działający system typu „Watchdog", zbierający informacje o stanie lokalnych procesów i Raportujący zebrane informacje do odpowiednio zarejestrowanych w procesie serwerów. Otwartą kwestią powiązaną z zarządzaniem zdalnymi zasobami i rozproszonymi lokalnymi procesami jest funkcjonalność przejmowania zarządzania nad aktywnym procesem przez dowolny ETLPE. W chwili obecnej system pozwala na wykonanie takiej operacji, ale nie została ona dodana, ponieważ temat ten powinien zostać rozszerzony o wyżej wymienione propozycje, które są pozostawione do dalszej pracy.

Zamierzamy również rozpatrywać dynamiczną realokację lokalnych i globalnych operatorów, tak aby system ETLPE elastycznie dostosowywał przetwarzanie strumienia do dynamicznych warunków panujących w znacznie rozproszonym środowisku (stąd założenie o operatorach nieblokujących, by ograniczyć liczbę danych w przypadku migracji).

Kolejnym aspektem, który został pominięty w tym artykule, jest bezpieczeństwo transmisji krotek. Zamierzamy wprowadzić dodatkowe operatory na wejściu i wyjściu lokalnego grafu transmisji, działające na zasadzie potwierdzenia przekazania krotki i ewentualną replikację procesu.

Wiele pracy wymaga również system rozproszonej kontroli procesu, który powinien z dowolnego ETLPE udostępnić pełny interfejs kontrolno-sterujący. W chwili obecnej pozwala na kontrolę procesu przez dostęp do poszczególnych (dla każdego ETLPE wchodzącego w skład procesu) logów prezentowanych w formie zasobów, zatrzymywanie, uruchamianie

i pobieranie informacji, dotyczących szczegółowych parametrów każdego z lokalnych operatorów.

LITERATURA

1. Abadi D. J., Ahmad Y., Balazinska M., Çetintemel U., Cherniack M., Hwang J. H., Lindner W., Maskey A. S., Rasin A., Ryvkina E., Tatbul N., Xing Y, Zdonik S.: The Design Of The Borealis Stream Processing Engine. CIDR Conference 2005.
2. Ahmad Y., Çetintemel U.: Network-Aware Query Processing for Stream-based Applications. 30th VLDB Conference, Toronto, Kanada, 2004.
3. Gorawski M., Marks P.: Distributed Stream Processing Analysis in High Availability Context. The First International Conference on Availability, Reliability and Security (ARES 2007), Austria, Wiedeń , 10-13. 04. 2007.
4. Gorawski M., Wolany P.: Integracja systemu rozproszonego przetwarzania strumienia danych z systemem agentów programowych. Konferencja Bazy Danych: Aplikacje i Systemy, 28.05. – 31.05.2007, Ustroń.
5. Ahuja R.K., Magnanti T.L., Orlin J.B.: Network Flow Theory, Algorithms, and Applications, Prentice-Hall, Englewood-Cliffs, New Jersey 1993.

Recenzent: Dr hab. Tadeusz Pankowski

Wpłynęło do Redakcji 6 listopada 2007 r.

Abstract

We present prototype system that supports designing and service of the ETL processes in distributed environment of data streams processing. Our solution is an alternative proposition which bases on flexible and scalable structure of equal processing nodes and network connections independent of physical network topology. Such network, the ETLPE, based on a distributed resources mechanism, allows automation of the widely distributed stream process design and ensures fully remote online control.

Adresy

Marcin GORAWSKI: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, Marcin.Gorawski@polsl.pl.
Wojciech KOŁODZIEJ: Politechnika Śląska, Instytut Informatyki, ul. Akademicka 16, 44-100 Gliwice, Polska, WKolodziej@polsl.pl.