MONITOROWANIE OBIEKTOWO ZORIENTOWANYCH APLIKACJI ROZPROSZONYCH, PODEJŚCIE ANSA[1]

Andrzej Król, Andrzej Uszok, Krzysztof Zieliński
Katedra Informatyki Akademi Górniczo-Hutniczej
Al. Mickiewicza 30, 30-059 Kraków
email: {genda, uszok, kz}@ics.agh.edu.pl

Streszczenie

Referat zawiera zwięzły opis systemu **LM** (Load Monitoring) monitorowania zewnętrznego obiektowych aplikacji rozproszonych. W kolejnych punktach przedstawiono model funkcjonalny systemu, zasadę jego implementacji w środowisku ANSA, sposób działania oraz korzystania z systemu.

Celem referatu jest przedstawienie implemenacji systemu **LM** (Load Monitoring) monitorowania zewnętrznego napisanego z uwzględnieniem specyfiki zarządzania obiektowo zorientowanymi systemami rozproszonymi.

Jako platformę implementacji tego systemu wybrano system ANSA [1, 2], tak więc rozważany system monitorowania stanowi sam obiektową aplikację rozproszoną. Punktem wyjścia dla prezentowanej implementacji systemu monitorowania jest jego model funkcjonalny omówiony krótko na początku referatu. Następnie podano podstawowe informacje odnośnie sposobu implementacji i korzystania z systemu. Ze względu na ograniczoną objętość referatu nie podano przykładu korzystania z systemu lecz ograniczono się tylko do omówienia ogólnych zasad programowania.

1 Model funkcjonalny systemu

Model funkcjonalny systemu monitorowania rozproszonych systemów komputerowych został zaproponowany w pracy [6] i obejmuje zbiór podstawowych funkcji odpowiedzialnych za generację, przetwarzanie, rozsyłania i prezentację informacji monitorowania. Dla monitorowania zewnętrznego gromadzona jest informacja o zachowaniu się obiektów lub systemu na podstawie zewnętrznych znamion ich działalności np.: przez zmianę obciążenia procesora, zajętości pamięci, średniego obciążenia komputera. Realizacja monitorowania zewnętrznego daje więc w rezultacie informację ogólną i nie odnoszącą się bezpośrednio do danego wybranego obiektu.

Dane monitorowania są generowane przez odpowiednie moduły systemu w postaci raportów stanów i zdarzeń. Raport stanu zawiera zbiór wartości wektora stanu oraz opcjonalnie inne informacje takie jak znacznik czasu lub identyfikator obiektu. Wektor ten reprezentuje stan obiektu w określonej chwili czasu i może być generowany w sposób następujący:

- Okresowo: raporty są generowane w oparciu o wcześniej zdefiniowaną regułę,

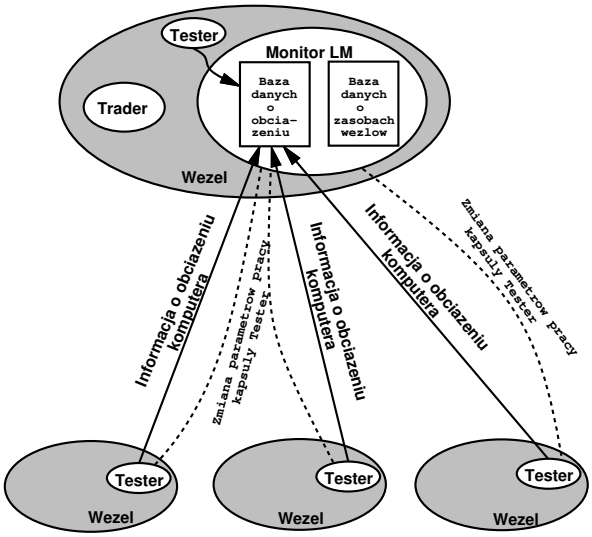
[1] Wykonano w ramach grantu KBN nr 8 S503 015 06

- Na żądanie: raport jest generowany po otrzymaniu polecenia wykonania tej operacji.
- Zdarzeniewo: raport jest generowany po wystąpieniu określonego zdarzenia.

Podstawowym założeniem odnośnie sposobu implementacji **LM** jest przyjęcie, iż system ten będzie stanowił zbiór kooperujących obiektów, których interfejsy będą dostępne z poziomu programu dowolnego użytkownika, realizujących rozważane wcześniej funkcje monitorowania. W tym sensie **LM** ma dostarczać publicznie dostępnej usługi monitorowania. Sprawia to, że implementacja tego systemu powinna stanowić obiektową aplikację rozproszoną napisaną w konwencji systemów otwartych [1, 4]. Narzuca to wybór platformy oprogramowania, nad którą powinien być zbudowany system monitorowania. Platformą spełniającą narzucone wymagania jest ANSA [1, 2, 3], a jej wybór do realizacji tej pracy był dodatkowo uzasadniony dostępnością tego systemu. Przyjęto koncepcję scentralizowanego modelu monitoringu to znaczy informacja na temat obciążenia poszczególnych węzłów systemu zbierana jest na jednym wyróżnionym komputerze. Stanowi to pewne ograniczenie systemu w zastosowaniu do bardzo dużych sieci. Istnieje jednak możliwość tworzenia *federacji* systemów monitorowania.

2 Opis implementacji systemu LM

Częściami składowymi systemu monitorującego **LM**, powstałego jako aplikacja systemu ANSA, są obiekty komunikujące się przy pomocy interfejsów. Strukturę systemu monitorowania przedstawiono na Rys. 1.



Rysunek 1: System monitoringu zbudowany w oparciu o środowisko ANSA.

Składa sie on z dwu typów kapsuł: **Monitor -** gromadzącej informacje o monitorowanym systemie oraz **Tester -** generujących wektory stanu. Wszystkie obiekty

systemu **LM** sa wielowątkowe. Umożliwia to współbieżną obsługę zdarzeń występujących w systemie. Przy opisie interfejsów obiektu zaznaczono liczbę procesów wirtualnych jakie zostały zainicjowane w danej kapsule. Poniżej zaprezentowano szczegółowy opis interfejsów i działania tych kapsuł.

Kapsuła **Monitor** posiada trzy interfejsy, z których każdy zawiera zbiór usług wykorzystywany przez używające go programy w określonym celu:

- **SlmDc** - System loading monitor - Data collecting

  Interfejs ten dostarcza operacji gromadzenia danych o obciążeniu komputerów. Jest on tworzony ze współczynnikiem współbieżności równym ilości węzłów w systemie. Oznacza to, że odbiór danych monitorowania z poszczególnych węzłów jest realizowany przez osobny wątek.

- **SlmAcd** - System loading monitor - Access to collecting data

  Interfejs ten oferuje funkcje umożliwiające programom dostęp do zebranych przez kapułę danych. Jest tworzony ze współczynnikiem współbieżności równym 4. Zdefiniowane w interfejsie funkcje umożliwiają uzyskanie informacji, albo o jednym komputerze, którego nazwę podano, lub o całym systemie.

- **SlmCp** - System loading monitor - Change parameters

  Interfejs ten oferuje funkcje umożliwiające zmianę parametrów działania monitora. Jest on tworzony ze współczynnikiem współbieżności równym 1. Funkcje tego interfejsu pozwalają zmieniać wartości zmiennych monitora: **Period** i **Delimiter** na nowe żądane wartości. Zmienne te posiadają następujące znaczenie:

  - **Period** - określa kwant czasu pomiędzy dwoma kolejnymi odczytami danych na temat obciążenia komputerów,
  - **Delimiter** - określa próg zmiany danych powodujący odświeżenie danych zapisanych w odpowiednim elemencie tablicy **Nodes** kapsuły **Monitor**, zawierającej informacje o węzłach.

Okres **Period** powinien być na tyle krótki aby system monitorowania reagował dostatecznie szybko na zmianę obciążenia. Z drugiej zaś strony zbyt częsty odczyt może spowodować nadmierne obciążenie systemu. Parameter progowy **Delimiter** pozwala na wstępna filtrację zbierania danych. Warunkiem przesyłania wyników odczytu monitorowanej informacji jest, aby względna zmiana wartości jednego z odczytywanych parametrów była większa niż zadana wartość progowa.

Kapsuła **Tester** posiada tylko jeden interfejs - **Tester**, którego funkcje umożliwiają zainicjowanie pracy tej kapsuły i wpływ na jej działanie. W interfejsie tym zdefiniowano 4 operacje. Pierwsza z udostępnianych operacji inicjuje proces zbierania przez **Tester** informacji o wektorze obciążenia. Jako argumenty tej operacji podawany jest przydzielony danemu komputerowi numer, oraz wartości inicjujace zmienne **Preiod** i **Delimiter**. Kolejna operacja umożliwia odczytanie aktualnych danych o wektorze stanu komputera. Dwie ostatnie operacje pozwalają zmieniać odpowiednio wartości parametrów **Period** i **Delimiter**.

3 Opis dzialania systemu

Kapsuła **Monitor** jest uruchamiana na samym początku pracy systemu **LM**. Wynikiem jej działania jest:

- Uruchomienie na każdym węźle tego systemu kapsuły **Tester**, zbierającej dane na temat obciążenia komputera, na którym działa.
- Inicjacja bazy danych zawierającej informacje na temat stanu komputerów systemu ANSA. Polega to na inicjacji elementów tablicy **Nodes** na podstawie pliku zawierającego dane na temat zasobów poszczególnych węzłów systemu.
- Wyeksportowanie referencji do interfejsów tak, aby mogły one być dostępne dla innych aplikacji ANSA.
- Instalacja funkcji pozwalającej na zakończenie systemu **LM**.

Po zakończeniu wykonywania procedury inicjacji, system monitorowania pracuje, a zbierane przez niego informacje są dostępne dla wszystkich aplikacji ANSA, które dokonują importu referencji interfejsu **SlmAcd** tej kapsuły. Sposób korzystania z systemu monitorowania z innych aplikacji niż napisanych w ANSA jest nieco bardziej złożony. Wymaga to korzystania z mechanizmów współpracy (*interoperability*) różnych systemów rozproszonych [8].

W kapsule **Tester** wykorzystano dwa sposoby uzyskiwania informacji o obciążeniu węzłów. Pierwszy z nich oparty jest na zbiorze parametrów zwracanych w strukturze **staststime** podczas wywołania systemowej funkcji **rstat**. Użycie jednak funkcji **rstat** ma kilka niedogodności:

- Funkcja ta jest ściśle związana z mechanizmami RPC, co nie zgadza się z założeniem napisania monitora opartego tylko na mechanizmach komunikacyjnych ANSA i wnosi dodatkowy narzut czasowy.
- Uzależnia program monitora od systemu SUNOS, w którym funkcja **rstat** występuje, a nie ma jej np. w systemie AIX komputerów IBM RS6000.

Dlatego w drugim rozwiązaniu zdecydowano sie zrezygnować z użycia tej funkcji na rzecz bezpośredniego odczytu danych z jądra systemu. Kapsuła **Tester** posiada specjalnie skonstruowaną funkcje **rrstat**, która odczytuje poszczególne dane z jądra i wypełnia strukturę **statstime**. Potrafi ona odczytać wszytkie potrzebne dane w systemie SUNOS i AIX.

4 Sposób wykorzystania systemu LM

Przedstawiona powyżej implementacja monitora **LM** stanowi pewien szkielet funkcjonalny. Podany zbiór interfejsów i operacji można traktować jako przykładowy i może on zostać bez trudu rozszerzony. Jedna z modyfikacji może dotyczyć dodania funkcji dostarczających informacje przydatne w równoważeniu obciążenia systemu (*load-balancing*). Zestaw takich operacji ma postać osobnego interfejsu eksportowanego przez kapsułę monitora i zawiera następujące funkcje:

- **BestNode** - funkcja ta zwraca nazwę najlepszego, według aktualnie posiadanych informacji, komputera wyszukanego w zbiorze węzłów systemu

ANSA przy pomocy algorytmu (*load balancing*) o nazwie będącej argumentem funkcji.

- **GetHostList** - funkcja ta zwraca listę nazw komputerów pracujących aktualnie w systemie ANSA. Może służyć do uzyskania przez program informacji o dostępnych zasobach sprzętowych systemu.
- **GoodNode** - funkcja ta zwraca listę nazw komputerów pracujących aktualnie w systemie ANSA, których obciążenie jest mniejsze od określonego przez argument jej wywołania.
- **NewGoodNodes** - funkcja ta bada jakie komputery, oprócz podanych w jej wywołaniu, są mniej obciążone niż określa to jeden z argumentów jej wywołania.
- **BadNodes** - funkcja ta bada, które z podanych komputerów są bardziej obciążone niż określa to argument jej wywołania.
- **StillGoodNode** - funkcja ta sprawdza czy obciążenie komputera o podanej nazwie jest mniejsze od podanej wartości.

Przedstawione oprogramowanie monitorowania dla sieci stacji roboczych może być łatwo przeniesione do nowocześniejszego środowiska CORBA (OMG) [7], mającego wiele cech wspólnych z systemem ANSA. Można bowiem w tym przypadku wykorzystać większość dotychczasowych rozwiązań bez większych zmian. Wykorzystanie środowiska CORBA umożliwia rozszerzenie proponowanego systemu monitorowania o wykorzystanie mechanizmów dostarczanych przez jej implementację ORBIX (IONA) [4] w postaci funkcji filtracji oraz dynamicznego wywołania interfejsu (Dynamic Invocation Interface).

Literatura

[1] ANSAware 4.0 --- Application Programmer's Manual. APM Ltd. Cambridge (1992).
[2] ANSAware 4.0 --- Systems Programmer's Manual. APM Ltd. Cambridge (1992).
[3] M. Bijałd, M. Gwóźdź, K. Zieliński, Rozproszone przetwarzanie w systemach otwartych - model ANSA, Informatyka 3 (1993).
[4] Draft Common Object Request Broker Architecture Revision 1.1. OMG report 91-12-1, OMG Inc., (1991).
[5] R.E. Johnson, B. Foote, Designing Reusable Clases. Journal of Object Oriented Programming, Jun/Jul (1988).
[6] M. Mansouri-Samani, M. Sloman, Monitoring Distributed Systems (A Survey) Imperial College Reseach Report No. DOC92/23 Apr. (1993).
[7] The Orbix Architecture. IONA technologies Ltd., (1993).
[8] K. Zieliński, G. Czajkowski, A. Uszok, Coordination through Interoperability in Distributed Processing, ECOOP'94, Bologna, 4-8 July, (1994){dostępny na ftp.uci.agh.edu.pl:/pub/papers/cs/interop.ps.gz}.