1996 ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ

Seria: INFORMATYKA z. 30 Nr kol. 1315

Andrzej KRÓL
Dominik MIODUNKA
Tomasz ROLA
Krzysztof ZIELIŃSKI

MIGRACJA W OTWARTYCH OBIEKTOWOWO ORIENTOWANYCH SYSTEMACH KOMPUTEROWYCH, PRZYKŁAD ANSA [1]

Streszczenie. W niniejszym artykule przedstawiono realizację rozszerzenia systemu ANSA o mechanizmy migracji obiektów. Własności tak zmodyfikowanego systemu przedstawiono na przykładzie realizacji obliczeń aplikacji.

MIGRATION IN OPEN OBJECT ORIENTED COMPUTER SYSTEMS, ANSA APPROACH

Summary. In this paper, the extension of ANSA with migration mechanisms has been described. The features of modified version of the system has been presented by the application computational example.

[1] Pracę wykonano w ramach grantu KBN nr 8 S503 015 06.

LA MIGRATION DANS LES SYSTÈMES OBJET ORIENTÉS OUVERTS, APPROCHE ANSA

Résumé. On présente une réalisation d'exention du système ANSA aux mécanismes de la migration des objects. Les propriétés de tels systèmes sont présent à l'aide de la réalisation d'une application exemplaire.

1. Wstęp

W systemie komupterowym, jaki stanowi zespół stacji roboczych połączonych lokalną siecią komputerową, bardzo często występuje zjawisko niepełnego wykorzystania oferowanych mocy obliczeniowych. Obciążenie poszczególnych węzłów jest zmienne w czasie, co dodatkowo komplikuje możliwość ich wykorzystania. Próbę dostosowania się do warunków panujących w takim systemie może stanowić przenoszenie obliczeń pomiędzy węzłami w celu adaptacji do zmieniającego się stanu systemu. W szczególności technika ta może dotyczyć obiektowych aplikacji rozproszonych.

W systemie obiektowym dostosowanie się do zmieniającego się obciążenia węzłów może polegać na migracji obiektów realizujących obliczenia. Udostępnienie mechanizmu migracji pozwala też na realizację obliczeń w tle, bez degradacji warunków pracy dla lokalnych użytkowników poszczególnych węzłów.

Celem niniejszego artykułu jest przedstawienie implementacji mechanizmu migracji w obiektowym systemie rozproszonym. Jako platformę implementacji wybrano system ANSA [1], tak więc system migracji sam też stanowi aplikację obiektową. Przy implemetacji systemu starano się maksymalnie wykorzystać specyficzne cechy systemu ANSA.

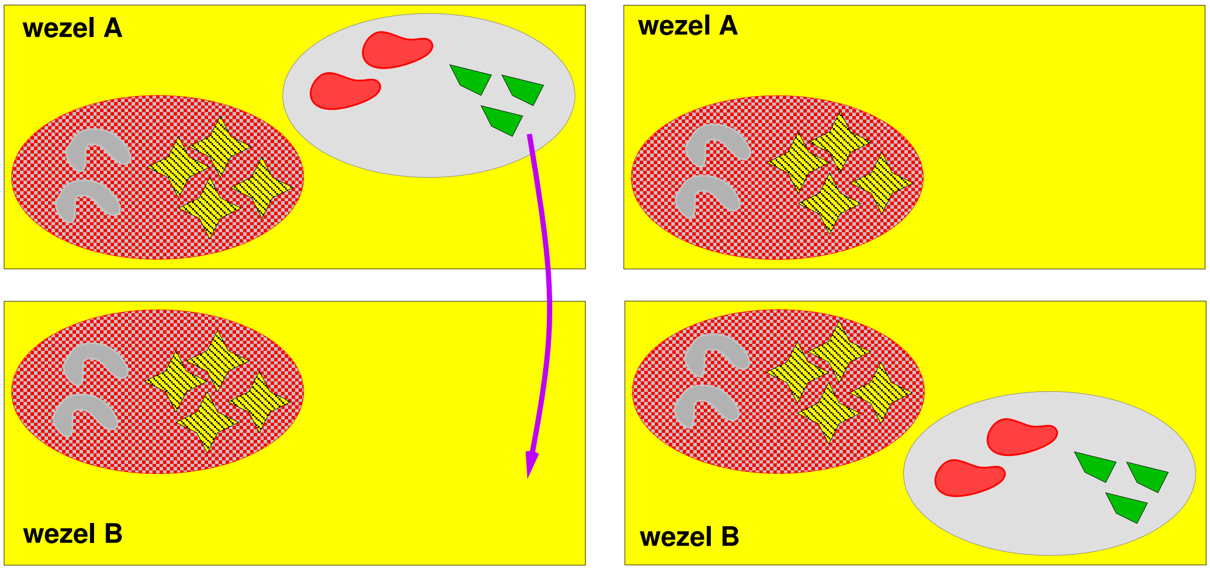
Punktem wyjścia dla prezentowanej implementacji systemu migracji jest opis proponowanego systemu migracji obiektów omówiony krótko w punkcie 2. Rezultaty eksperymentów ze zrealizowanym systemem zaprezentowano w punkcie 3.

2. System migracji obiektów

Elementy środowiska ANSA nie posiadają wbudowanych cech umożliwiających ich przemieszczanie. Możliwość przemieszczania obiektów pomiędzy węzłami wymaga nadania im specjalnej cechy migrowalności. Mechanizm migracji może dotyczyć tylko tych elementów środowiska, którym nadano tę cechę.

2.1. Koncepcja systemu

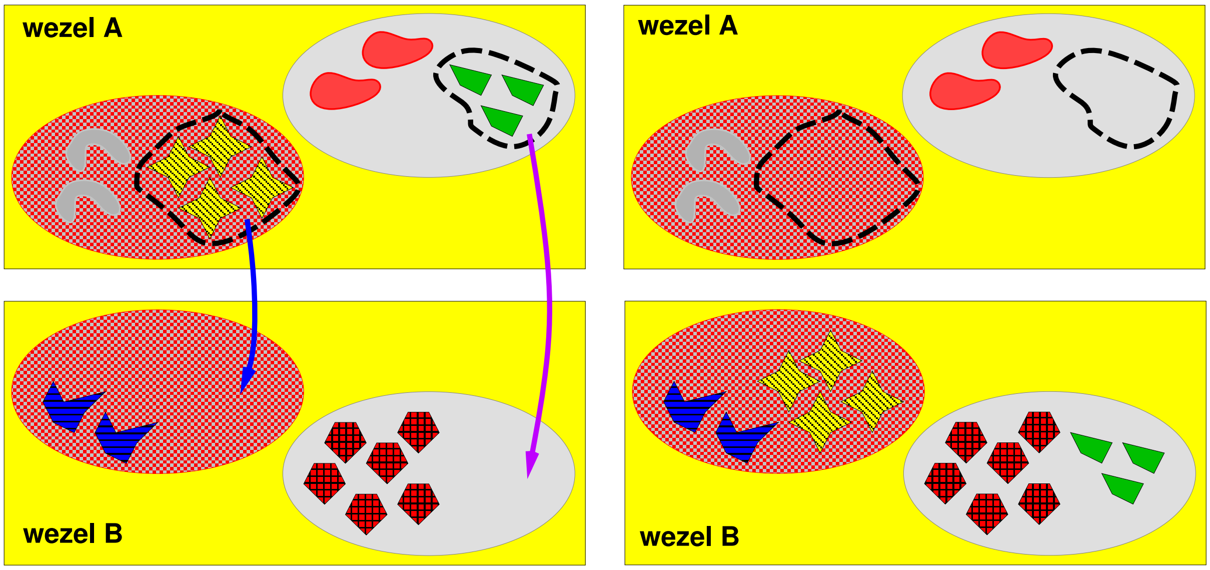
Realizacja migracji w przypadku ogólnym jest bardzo złożonym zadaniem. Ponieważ zasadniczą część obliczeń w obiektowej aplikacji rozproszonej realizują obiekty serwery, przyjęto więc założenie, że opracowany mechanizm migracji będzie je obejmował. Proponowany mechanizm migracji dla systemu ANSA może dotyczyć dwu poziomów: migracji kapsuł razem ze najdującymi się w nich obiektami lub migracji obiektów pomiędzy kapsułami tego samego typu. Wprowadzono więc dwa następujące pojęcia: migracja gruboziarnista i migracja drobnoziarnista. W przypadku migracji gruboziarnistej przemieszczaniu podlegają obiekty razem z kapsułami w, których one znajdują się (rysunek 1).



Rys. 1. Migracja gruboziarnista
Fig. 1. Coarse-grain migration

Pomysł migracji drobnoziarnistej polega na wyeliminowaniu dość kosztownego procesu tworzenia nowych kapsuł podczas migracji. Określa się klaster kapsuł tego samego typu, pomiędzy którymi można dokonywać migracji obiektów. Takiemu rodzajowi migracji może podlegać zarówno pojedyńczy obiekt, jak też grupa obiektów tworząca klaster. W drugim przypadku podczas pojedyńczego procesu migracji przemieszczany jest cały klaster obiektów. Migracja drobnoziarnista realizowana jest podobnie do zaprezentowanej powyżej migracji gruboziarnistej, przy czym najistotniejszą różnicę stanowi brak etapu tworzenia nowej kapsuły (rysunek 2).

Jako uzupełnienie zaprezentowanych powyżej mechanizmów migracji przewidziano możliwość blokowania migracji dla wybranych obiektów. Użytkownik, w razie wystąpienia takiej potrzeby, może wskazać obiekty, które nie powinny podlegać migracji. Taka blokada może dotyczyć wybranego kwantu czasu. Po tym okresie obiekt może ponownie podlegać migracji. Daje to użytkownikowi dodatkowe możliwości sterowania przebiegiem wykonania aplikacji.



Rys. 2 Migracja drobnoziarnista
Fig. 2 Fine-grain migration

2.2. Modyfikacja interfejsu

Możliwość korzystania z mechanizmu migracji wymaga zmian w definicji interfejsu obiektu. Dotyczy to sposobu nadania danemu obiektowi cechy migrowalności.

Przy definincji interfejsu w języku IDL należy zapewnić, aby był on kompatybilny z typem `Snapshot`. Polega to na dodaniu w ciele definicji wyrażenia `IMPLEMENTATION IS COMPATIBLE WITH Snapshot`. Definicja typu `Snapshot` jest następująca:

```
Snapshot : INTERFACE =
BEGIN
 StateSnapshot : TYPE = SEQUENCE OF OCTET;
 ObjectInterfaces : TYPE = SEQUENCE OF ansa_InterfaceRef;
 Snapshot : TYPE = RECORD [
           sSnap           : StateSnapshot,
           interfaces      : ObjectInterfaces,
           oMyRef          : ansa_InterfaceRef,
           oCapsule        : ansa_InterfaceRef,
           cTerminated     : ansa_InterfaceRef,
           oTemplate       : STRING,
           oArguments      : STRING,
           oEnvironment    : STRING,
           oArguments      : STRING,
           oEnvironment    : STRING,
           cPath           : STRING,
           cTemplate       : STRING,
           cArguments      : STRING,
```

```
           cEnvironment   : STRING,
           cData          : CARDINAL
          ];
MakeSnapshot : INTERROGATION OPERATION [] RETURNS [Snapshot];
InstallSnapshot : INTERROGATION OPERATION [S : Snapshot] RETURNS [BOOLEAN];
FreezeObject : INTERROGATION OPERATION [] RETURNS [];
IsFrozen : INTERROGATION OPERATION [] RETURNS [ BOOLEAN ];
DeFreezeObject : INTERROGATION OPERATION [] RETURNS [];
AddObjectInfo : INTERROGATION OPERATION [
               oMyRef         : ansa_InterfaceRef;
               oCapsule       : ansa_InterfaceRef;
               oTemplate      : STRING;
               oArguments     : STRING;
               oEnvironment   : STRING
              ] RETURNS [];
AddCapsuleInfo : INTERROGATION OPERATION [
               cPath          : STRING;
               cTemplate      : STRING;
               cArguments     : STRING;
               cEnvironment   : STRING;
               cTerminated    : ansa_InterfaceRef;
               cData          : CARDINAL
              ] RETURNS [];
END.
```

W rekordzie typu `Snapshot` przechowywana jest pełna informacja potrzebna do zapamiętania stanu obiektu. Poszczególne pola mają następujące znaczenie:

- `sSnap` - jest to obraz stanu obiektu zapisany w postaci ciągu bajtów,
- `interfaces` - opis wszytkich interfejsów udostępnianych przez obiekt w czasie jego powstawania,
- pozostałe pola służą do przechowywania informacji o obiekcie i kapsule, w której on się znajduje. Są one niezbędne do odtworzenia kapsuły i obiektu w nowym miejscu.

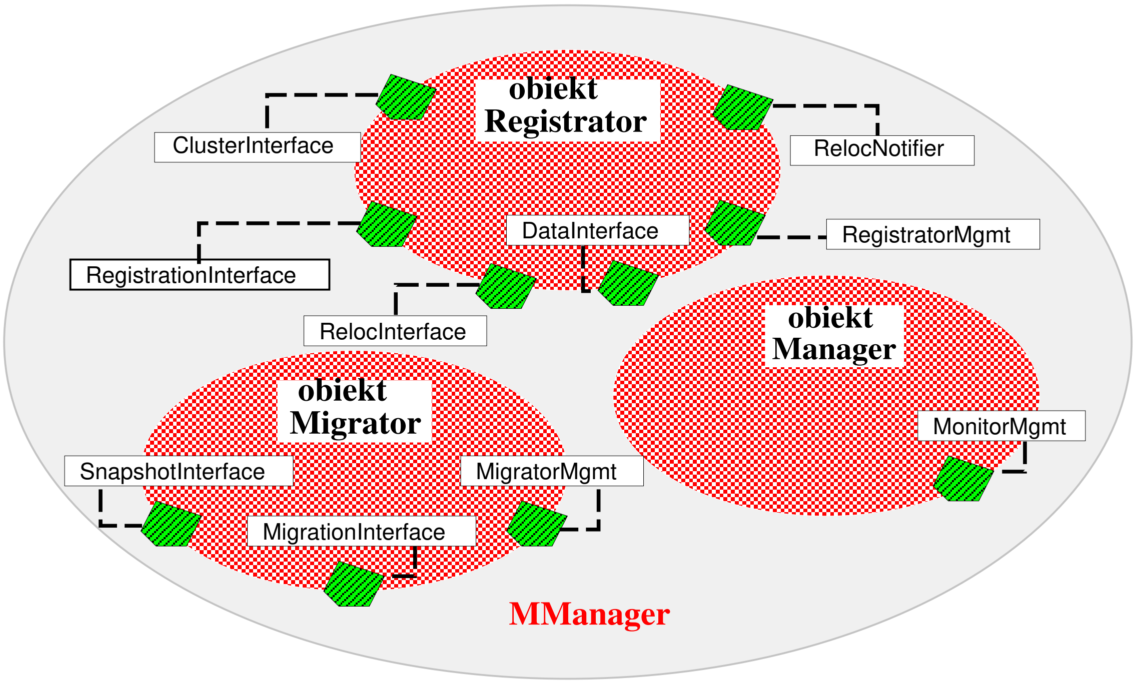
Operacje `MakeSnapshot` i `InstallSnapshot` służą do wykonywania i instalowania obrazu stanu. Operacje `MakeSnapshot` i `InstallSnapshot` służą do zapamiętywania i instalowania obrazu stanu obiektu. Przed ich wywołaniem należy zamrozić stan obiektu za pomocą operacji `FreezeObject`. Osiąganie stanu zmrożenia obiektu polega na wykonaniu aktualnie realizowanych zleceń i blokadzie przyjmowania nowych. W stanie zamrożenia obiekt nie posiada więc aktywnych procesów lub wątków. Stan obiektu można odmrozić wywołując operację `DeFreezeObject`. Do sprawdzenia aktualnego stanu obiektu służy operacja `IsFrozen`. Operacje `AddCapsuleInfo` oraz `AddObjectInfo` są używane do umieszczenia w obrazie stanu informacji dotyczących kreacji obiektu i jego kapsuły macierzystej.

2.3. Struktura systemu

Zasadniczy element systemu migracji stanowi kapsuła `MManager`, zarządzającą migracją kapsuł i obiektów. Kapsuła ta posiada własne niezależne od `Tradera` systemu ANSA struktury danych. Zarejestrowane są w nich po utworzeniu wszystkie obiekty, które mogą podlegać migracji.

Kapsuła ta zawiera niezbędne dane i informacje do realizacji procesu migracji i zarządzania jego wykonaniem. Przyjęto, że wewnętrzną strukturę kapsuły `MManager` stanowić będą trzy współpracujące ze sobą obiekty. Przyjęcie takiego rozwiązania spowodowane było chęcią eliminacji problemów związanych z synchronizacją dostępu do globalnych danych kapsuły. W kapsule `MManager` tworzone są następujące obiekty (rysunek 3):

- `Manager` - pobiera dane o stanie systemu z modułu monitorowania. Na podstawie tych danych podejmuje decyzję o ewentualnej migracji.
- `Migrator` - przeprowadza kolejne etapy procesu migracji na żądanie obiektu monitora,
- `Registrator` - zarządza referencjami do interfersów oferwowanych przez obiekty serwerów usług. Obejmuje to rejestrację obiektów, akutalizacje struktur danych relokacji, udostępnianie bieżących danych o referencji do interfejsu i wyrejestrowywanie.



Rys. 3. Budowa kapsuły `MManager`
Fig. 3. `MManager` capsule structure

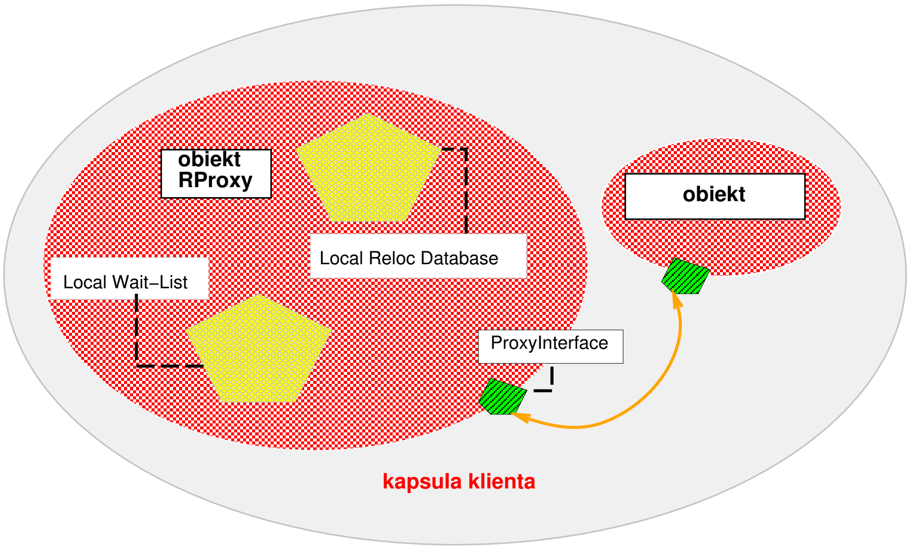
Przeznaczeniem obiektu `Manager` jest zarządzanie procesem migracji. Obiekt ten pobiera dane z kapsuły `Mastermon` i na ich podstawie podejmuje decyzje o zleceniu wykonania migracji obiektu. Operacja ta zlecana jest do wykonania obiektowi `Migrator`. Obiekt `Manager` udostępnia interfejs `MonitorMgmt` dla celów zarządzania.

Obiekt `Migrator` realizuje proces migracji dla wybranego obiektu. Obejmuje to zamrożenie i zapamiętanie stanu obiektu, a następnie odtworzenie obiektu na innym węźle. Obiekt `Migrator` dostarcza następujące interfejsy: `MigratorMgmt`,`SnapshotInterface` i `MigrationInterface`. Interfejs `MigrationMgmt` zapewnia operację umożliwiającą pobranie referencji udostępnianych przez obiekt `Migrator` interfejsów. Kolejny interfejs `SnapshotInterface` przeznaczony jest do operacji związanych z zapamiętaniem i odtwarzaniem stanu obiektu. W interfejsie tym umieszczono podzbiór operacji interfejsu `Snapshot`. Ostatni z dostarczanych Interfejsów `MigrationInterface` udostępnia operacje umożliwiające migrację kapsuły i obiektu, a także migrację obiektu w obrębie klastra kapsuł.

Obiekt `Registrator` przechowuje i udostępnia informacje o kapsułach i obiektach zarejestrowanych w systemie. Tylko zarejestrowane obiekty mogą podlegać procesowi migracji. Obiekt ten udostępnia następujące interfejsy: `RegistratorMgmt`, `DataInterface`, `RegistrationInterface`, `RelocInterface`, `RelocNotifier` i `ClusterInterface`. Interfejs `RegistratorMgmt` udostępnia operacje służące do pobrania referencji uodstępnianych przez obiekt Registrator interfejsów. Interfejs `DataInterface` udostępnia operacje, które umożliwiają pobranie z obiektu `Registrator` informacji o zarejestrowanych w jego strukturach danych kapsułach oraz obiektach. Interfejs `RegistrationInterface` udostępnia operacje, dzięki którym obiekt aplikacji może zarejestrować swój interfejs (typu `Snapshot`) w kapsule `MManager`. Następne dwa interfejsy dostarczają operacji związanycm z obsługą referencji. Interfejs `RelocInterface` udostępnia operacje umożliwiające rejestrację, znajdowanie i wyrejestrowywanie referencji do interfejsów. Kolejny interfejs `RelocNotifier` udostępnia operacje umożliwiające stosowanie mechanizmu powiadamiania o zmianie lokalizacji interfejsu. Dzięki temu klient posiadający referencję do starej lokalizacji interfejsu może odnaleźć interfejs po zmianie lokalizacji. Kolejny interfejs `ClusterInterface` dostarcza operacji niezbędnych przy tworzeniu klastrów obiektów.

Przy migracji obiektów serwerów występuje konieczność powiadamiania obiektów klientów o występującej zmianie referencji oferowanych przez serwery interfejsów. W tym celu zaproponowano następujący mechanizm. Klient przed wywołaniem operacji na jakimś interfejsie powinien sprawdzić czy nie istnieje jego nowsza wersja. Tworzy się więc w kaspule klienta lokalną reprezentację obiektu `Registrator` o nazwie `RProxy`. Obiekt ten tworzony jest na życzenie klienta. Obiekt `RProxy` posiada następujące struktury danych (rysunek 4):

- `LocalWaitList` - jest kopią centralnej `WaitList` posiadanej przez obiekt `Registrator`, spełnia podobne zadania jak lista centralna,
- `LocalRelocDatabase` - jest kopią centralnej struktury danych relokacji obiektu `Registrator`, umożliwia otrzymanie nowej referencji interfejsu.



Rys. 4. Budowa kapsuły klienta zawierająca obiekt `RProxy`
Fig. 4. Client capsule structure with `RProxy` object

Rozwiązanie to jest też bardzo interesujące w przypadku, gdy przy dużej liczbie obiektów klientów korzystających z usług obiektów serwerów, mogących podlegać migracji, wąskim gardłem może być dostęp interfejsu do obiektu `Registrator` kapsuły `MManager`. Wszytkie żądania podania aktualnej referencji musiałyby być sekwencyjnie realizowane przez ten obiekt, co powodowałoby określone narzuty czasowe, zwłaszcza związane z przeszukiwaniem jego struktur danych. Wprowadzenie mechanizmów związanych z użyciem obiektu lokalnego przedstawiciela eliminuje ten problem, gdyż obiekt `Registrator` odpowiedzialny jest tylko za powiadamianie lokalnych obiektów `RProxy`. Odwołania mające na celu pobranie aktualnej referencji są realizowane teraz lokalnie i dla różnych węzłów mogą być obsługiwane równocześnie. Zwiększa to potencjalnie możliwość skalowalności systemu. Takie rozwiązanie elimiuje też część narzutów na komunikację z obiektem `Registrator` kapsuły `MManager`.

3. Przykład obliczeń

Badania opracowanego systemu przeprowadzono na aplikacji typu *farmer–worker*. Występują w niej dwa podstawowe rodzaje kaspuł, a mianowicie kaspuły `Worker` realizujące obliczenia oraz kapsuły `Client` zlecające ich wykonanie. Kapsuła `worker` dla przeprowadzania obliczeń udostępnia interfejs `Work`.

Interfejs ten uzupełniony o niezbędne elementy do realizacji migracji, przyjmuje następującą postać:

```
Work : INTERFACE =

IMPLEMENTATION IS COMPATIBLE WITH Snapshot

BEGIN
DoComputing : INTERROGATION OPERATION [ num : CARDINAL ] RETURNS [];
Outputs : INTERROGATION OPERATION [ num : CARDINAL ] RETURNS [CARDINAL];
Lock : INTERROGATION OPERATION [ num : CARDINAL ] RETURNS [BOOLEAN];
CheckLock : INTERROGATION OPERATION [ num : CARDINAL ] RETURNS [BOOLEAN];
Free : INTERROGATION OPERATION [ num : CARDINAL ] RETURNS [BOOLEAN];
END.
```

Celem przeprowadzonych testów było oszacowanie czasu potrzebnego na realizację procesu migracji. Badania przeprowadzono na zespole 4 stacji roboczych SUN (2 SPARCstation 2 i 2 SPARCstation IPC). Mierzony był całkowity czas wykonania operacji migracji. Używano w tym celu udostępnianej przez środowisko ANSA funkcji `get_time` równoważnej systemowej funkcji `gettimeofday`. Dla potrzeb realizacji tego testu migracja polegała na kolejnym przemieszczaniu obiektu pomiędzy węzłami. Zaprezentowane poniżej czasy stanowią średnią arytmetyczną uzyskanych wyników. Średnie czasy otrzymano poprzez wielokrotne (1000 pomiarów) powtarzanie przebiegu testu.

Badane były dwa przypadki realizacji procesu migracji: migracja obiektów razem z kapsułami (migracja gruboziarnista) i migracja na poziomie obiektów w obrębie klastra kapsuł (migracja drobnoziarnista).

W pierwszej kolejności przetestowano wariant migracji gruboziarnistej. Dla badanej aplikacji czas zapamiętywania stanu obiektu wynosił 194[ms]. Tworzenie kapsuły w nowym miejscu zajmowało 340[ms], tworzenie zaś obiektu w tej kapsule 39[ms]. Odtwarzanie stanu obiektu w nowej kapsule trwało 225[ms]. Całkowity czas realizacji procesu migracji pojedynczej kapsuły dla badanej aplikacji był rzędu 1.54[s]. Znajdowanie nowej relokacji do interfejsu obiektu wewnątrz migrowanej kapsuły zajmowało 16[ms].

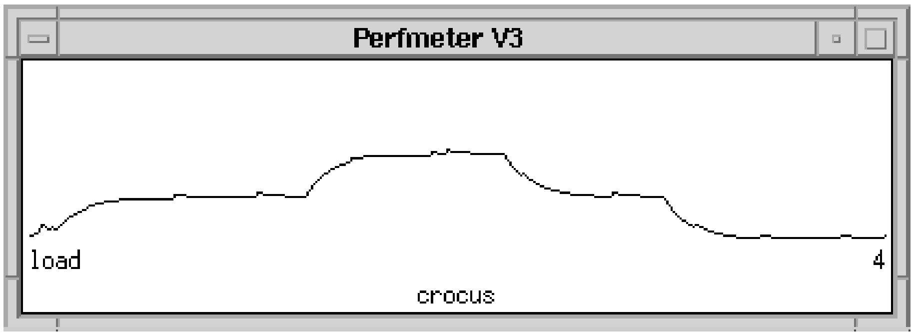
Testy dla migracji drobnoziarnistej przeprowadzono w następujący sposób. Na wszystkich węzłach systemu utworzono kapsuły badanej aplikacji. Utworzone kapsuły zostały zarejestrowane jako należące do jednego klastra. W wybranej kapsule utworzone zostały dwa obiekty serwera, w pozostałych trzech po jednym. Migracji podlegał pojedyńczy obiekt z kapsuły dwuobiektowej, traktowany jako klaster obiektów. Pozostałe obiekty serwerów w testowanej aplikacji tworzą własne klastry obiektów. Taka konfiguracja eksperymentu miała na celu proste określenie czasu migracji związanego tylko z poziomem obiektów, bez konieczności wykonywania operacji związanych z poziomem kapsuł.

Poszczególne fazy procesu migracji przebiegały podobnie do migracji gruboziarnistej. Czas zapamiętywania stanu obiektu wynosił 173[ms]. Tworzenie kopii obiektu po przeniesieniu do innej kapsuły trwało 31[ms]. Odtwarzanie zaś stanu obiektu podlegającego migracji zajmowało 205[ms]. Przy realizacji migracji drobnoziarnistej średni czas realizacji całego procesu wynosił 1.01[s]. Jak widać, zysk w czasie wykonania procesu migracji dotyczy głównie eliminacji kosztownego procesu tworzenia nowej kapsuły.

Oszacowany powyżej rząd wielkości czasu potrzebnego na realizację procesu migracji może stanowić podstawę do określenia rodzajów aplikacji, dla których mechanizm ten może mieć praktyczne zastosowanie. Interesujące jest więc pokazanie, jaki wpływ na przeprowadzenie obliczeń może mieć praktyczne zastosowanie mechanizmu migracji.

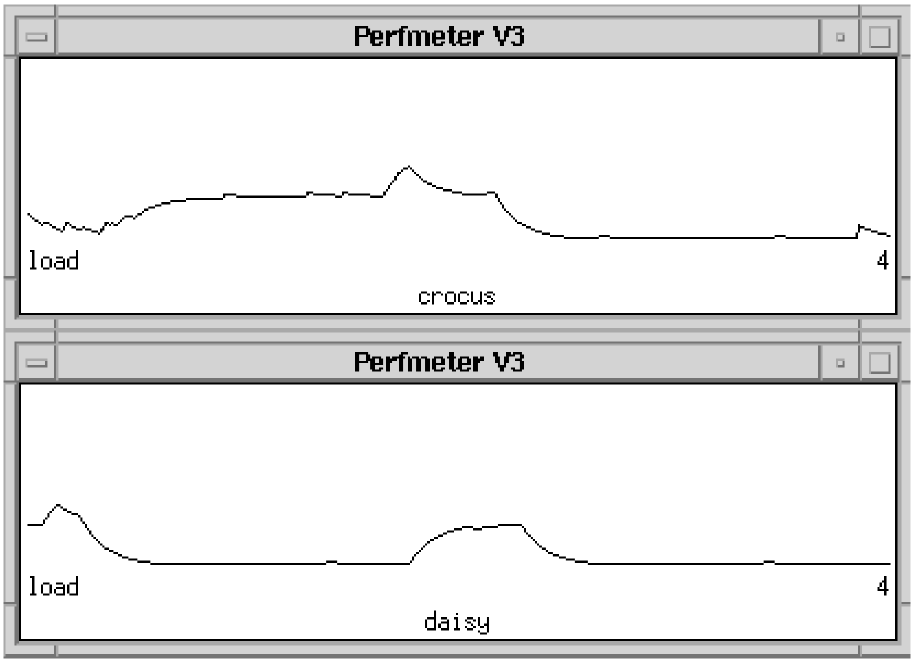
Celem jaki sobie postawiono przy korzystaniu z mechanizmu migracji jest nie chęć przyspieszenia obliczeń, co jest możliwe, ale racjonalne wykorzystanie wolnych w danym momencie zasobów środowiska rozproszonego. Sprowadza się to do realizacji obliczeń, tak aby lokalni użytkownicy mieli możliwość wykonywania własnych zadań ma bieżąco bez ograniczeń.

Na rysunku 5 przedstawiono realizację obliczeń równocześnie z wykorzystywaniem danego węzła przez lokalnego użytkownika. Prezentowany rysunek przedstawia przebieg zmian obciążenia węzła w czasie realizacji obliczeń. Widoczne jest wyraźne sumowanie się obciążeń generowanych przez aplikację i zadanie użytkownika. W konsekwencji występuje zwiększenie obciążenia stacji roboczej do poziomu mogącego utrudniać realizację zadań lokalnego użytkownika. Realnym rozwiązaniem tego problemu jest przeniesienie obliczeń na inny węzeł systemu.



Rys. 5. Obliczenia realizowane bez migracji obiektu aplikacji
Fig. 5. Computation without application object migration

Na kolejnym rysunku 6 zaprezentowano zmianę poziomów obciążenia węzłów dla aplikacji rozproszonej, przenoszącej obliczenia (dokonuje się migracji obiektu realizującego obliczenia) z węzła obciążonego przez lokalnego użytkownika na węzeł w danym momencie nieobciążony. Po przekroczeniu zadanego progu obciążenia stacji roboczej podejmowana jest decyzja o migracji obiektu aplikacji na nieobciążony w danej chwili węzeł. W wyniku tej akcji otrzymuje się widoczne na rysunku zmniejszenie obciążenia węzła. Szczegółowe wyniki badań zostały przedstawione w pracy [2], [6].



Rys. 6. Obliczenia realizowane z migracją obiektu aplikacji na nieobciążony węzeł
Fig. 6. Computation with application object migration

4 Podsumowanie

Przeprowadzone badania wykazały możliwość i celowość uzupełnienia systemu ANSA o mechanizm migracji obiektów.

Uzyskane wyniki eksperymentalne mają raczej charakter jakościowy, a nie ilościowy, gdyż te ostatnie dotyczą raczej konkretnej aplikacji. Inny też był cel przeprowadzonych eksperymentów. Przeprowadzone pomiary miały bardziej na celu określenie przeciętnych kosztów użycia tych mechanizmów przez aplikację. Na tej podstawie można określić klasę aplikacji, gdzie mechanizmy te mogą mieć zastosowanie.

Znacznie większe znaczenie posiadają ogólne cechy powstałej rozszerzonej architektury otwartych obiektowo orientowanych systemów komputerowych i przyjęta metodologia tworzenia implementacji dla konkretnego środowiska, którą można przenieść na inne środowiska rozproszone.

Badany system zachowuje się zgodnie z przewidywaniami, reagując prawidłowo na zmianę obciążenia i odpowiednio adaptuje przebieg wykonania aplikacji.

Jak wynika z przeprowadzonych testów, średni czas przeprowadzania procesu migracji jest rzędu pojedyńczych sekund. Czas ten jest w pełni akceptowalny z punktu widzenia lokalnego użytkownika węzła. Uzyskane wyniki pozwalają też wysnuć pewne wnioski co do budowy aplikacji korzystającej z opracowanych mechanizmów. Aby zapewnić odpowiednio szybką reakcję na występujące w systemie zmiany obciążenia poszczególnych węzłów, obiekty serwery aplikacji nie powinny wykonywać jednorazowo obliczeń znacząco dłuższych od czasu samej migracji. Niespełnienie tego postulatu może spowodować znaczące opóźnienia w przemieszczaniu obiektów pomiędzy węzłami. Spowoduje to odczuwalne przez użytkownika przeciążenie stacji roboczej. Zjawisko to jest niekorzystne i należy jego unikać.

LITERATURA

[1] ANSAware 4.0 — System Programmer's Manual. APM Ltd. Cambridge 1992.

[2] Król A.: Mechanizmy równoważenia obciążenia w obiektowo orientowanych otwartych komputerowych systemach rozproszonych. Rozprawa doktorska, AGH Kraków, Katedra Informatyki 1995.

[3] Król A., Uszok A., Zieliński K.: Równoważenie obciążenia w obiektowych systemach rozproszonych podejście ANSA. ZN s. Informatyka, z. 28, Politechnika Śląska, Gliwice, 1995.

[4] Król A., Uszok A., Zieliński K.: Monitorowanie obiektowo zorientowanych aplikacji rozproszonych, podejście ANSA. Konferencja Informatyka na wyższych uczelniach dla gospodarki narodowej, Gdańsk, 17-19 listopad 1994.

[5] Uszok A., Król A., K. Zieliński K.: Cluster Management Software for Open Object Oriented Systems. HPCN94, Münich, Lecture Notes in Computer Science 797, Springer-Verlag, 1994.

[6] D. Miodunka, T. Rola: Migracja obiektów w systemach rozproszonych. Praca magisterska, AGH Kraków, Katedra Informatyki 1995.

Recenzent: Dr hab. inż Tadeusz Czachórski

Wpłynęło do Redakcji 21 grudnia 1995 r.

Abstract

In this paper, the extension of ANSA with migration mechanisms has been described. An implementation of the system enriches the orginal ANSA model functionality with LM (Load Monitoring) and Migration modules and interfaces. Two types of migration strategy have been proposed, e.g.: fine-grain and coarse-grain. The methodology of the system extension usage by an application programmer has been specified. Some preliminary evaluation data of the migration software performance study have been discussed. The implemented migration software has been tested for some typical applications.