# Czynniki wpływające na równoważenie obciążenia

* Środowisko – architektura procesorów, typ współdzielonych zasobów oraz forma i typ połączeń pomiędzy procesorami.
* Charakter obciążenia
* Dostępne narzędzia do rozkładania obciążeń – jest to najważniejszy czynnik reprezentowany przez procedury i programy odpowiedzialne za rozkład obciążenia.

Z tego powodu wymagane są dwa typy narzędzi: Informacji oraz przetwarzania. Narzędzia informacji określają położenie procesu, natomiast narzędzia przetwarzania przenoszą procesy pomiędzy procesorami w systemie oraz zapewniają dostęp do zasobów systemowych.

# Realizacja obliczania obciążenia

Algorytm oblicza liczbę procesów, które są gotowe do uruchomienia, ale czekają na CPU i liczbę obecnie wykonywanych procesów przez CPU. Pojawia się tu jednak problem kiedy obciążenie zmienia się znacznie w małych przedziałach czasu. Najbardziej popularną metodą wykorzystywaną w przypadku takich problemów jest branie średniej co określony przedział czasu.

Wi - Procesy gotowe do wykonania

(ti-1, ti), i = 1,2 ..., µ – przedział czasu

T – jednostka czasu rozkładania obciążenia

Przybliżenie obciążenia w przedziale czasowym T

Ta metoda została zaimplementowana w systemie MOSIX

# Rozpowszechnianie informacji o obciążeniu

Występują dwie implementacje rozpowszechniania informacji pomiędzy węzłami. W niektórych implementacjach odpowiada za to jeden węzeł, a w innych każdy węzeł klastra wymienia informacje z innymi samodzielnie.

(Rysunek)

W klastrze stacji roboczych równoważenie obciążenia może być jedynie efektywne przy wykorzystaniu wiedzy o stanie indywidualnych węzłów wokół klastra. Innymi słowy wiedza ta jest wykorzystywana do optymalnego równoważenia rozprowadzania obciążenia. Algorytm odpowiedzialny za zbieranie informacji oraz ich rozpowszechnianie decyduje w jaki sposób te informacje trafią do zarządcy procesów. Celem zbierania informacji jest pomoc w podejmowaniu decyzji w czasie rzeczywistym.

W celu wymiany informajci w systemach zdecentralizowanych stosuje się mechanizmy: broadcast, multicast, prawdopodobieństwa. W mechaniźmie broadcast każdy węzeł cyklicznie wysyła informacje do wszystkich. Mechanizm multicast polega na wysyłaniu informacji przez węzeł do określonej grupy multicast. Ogranicza to znacznie generowany ruch w sieci. Mechanizm opierający się na prawdopodobieństwie ma na clu minimalizację wiadomości pomiędzy węzłami poprzez losowe wysyłanie wiadomości do określonej ilości węzłów.

# Decyzje o migracji

Algorytm migracji jest odpowiedzialny za decyzję o tym, który proces wymaga migracji i gdzie ma zostać wykonany. Mechanizm wybiera procesor na którym zostanie wykonany proces. W idealnym scenariuszu wybrany procesor powinien mieć minimalny koszt wykonania by zapwnić poprawe w systemie zarządzania obciążeniem. Do podjęcia takiej decyzji różne czynniki powinny być wzięte pod uwagę jak: czas odpowiedzi, wielkość tablicy procesu, wielkość wolnej pamięci, typ procesu oraz migracja. Celem jest poprawa wydajności poprzez zlecenie wykonanie zadania najmniej obciążonej jednostce.

Migracja ma miesce kiedy tablica procesu wewnąrz systemu operacyjnego jest niemal pełna. W takim przypadku tworzony jest nowy proces nazywany „dzieckiem”, jest to tak zwana częściowa migracja. Dodatkowo jeśli nie ma miejsca w węźle do stworzenia nowego procesu, wtedy algorytm tworzy proces na innym porcesorze. Typ procesu odgrywa ważną rolę. W większości implementacji wywołanie systemowe fork() tworzy process “dziecko” które może być przenoszone do innego węzła, przy czym jest zwyczaj do przenoszenia również wątka procesu. Jednakże czas i rozmiar mają największe znaczenie podczas migracji. Jeśli wielkość lub czas procesu „dziecka” jest niski – nie ma sensu migracji. Ma na to wpływ sam czas migracji. W zależności od systemu mechanizm ten wykonywany jest albo przez jeden wątek, albo jest on rozproszony na cały system.

# Rodzaje implementacji

Istnieją dwie implementacjeSSI równoważąch obciążenie. Podział polega na ingerencji w system operacyjny. Niektóre implementacje ingerują w system inne nie.

Freedman – migracja procesów poprzez migrację jedynie obrazu pamięci.

Condor – zapewnia zarządzanie procesami oraz migrację. Jest on zaimplementowany jedynie w przestrzeni urzytkownika i nie wymaga modyfikacji jądra systemu.

Głównym celem jest identyfikacja bezczynnych węzłów w środowisku i zlecenie pracy na nich. Wymaga to modyfikacji wykonywanego programu i podlinkowania bibliotek migracyjnych Condora podczas kompliacji.

Ze względów wydajnościowych jak i dla wygody użytkownika powstały systemy SSI stworzone od podstaw jak Accent, Amoeba, Angel, GLUnix. Problemem systemów budowanych od podstaw jest ich niestandardowość. Są to systemy zaprojektowane pod kątem specyficznych zadań.

Przeprowadzone badania dowiodły, że inteligentne rozmieszczanie procesów przynosi wymierne korzyści. Kluczem jest wykorzystanie informacji o węzłach klastra i procesach. Tych cech nie posiada standardowy system operacyjny.Pierwszym OS posiadającym tę funkcjonalność był MOSIX. Posiada on zbiór modyfikacji i dodatków do jądra Linuxa. Podstawowymi cechami są: równoważenie obciążenia i transparentne migracje procesów. Równoważenie obciążenia jest ciągle osiągane w trakcie pracy. Jeśli wymagania procesu przekraczają próg, proces będzie przeniesiony do innego mniej obciążonego węzła. Wektor obciążenia jest trzymany w każdym węźle i posiada informacjeo obciążeniach innych losowo wybranych węzłów.

# OpenMOSIX

OpenMOSIX wywodzi się ze skomercjalizowanego w 2001 projektu MOSIX. Posiada on ograniczenia nie pozwalające na migracje procesów współdzielonej pamięci. W konsekwencji aplikacje które wykorzystują pamięć współdzieloną nie czerpią korzyści z korzystania z niego. Projekt OpenMOSIX został zwieszony w 2008. Jego kontynuacją jest LinuxPMI.

Mechanizm równoważenia obciążenia dązy do równoważenia procesów na procesorze woków węzła poprzez migrację nadmiarowych. W tym przypadku została wprowadzona delegacja wewnątrz jądra na podobieństwo wątków. Ta delegacja trzyma informacje o migracji procesów. Wynikiem tego uruchomiony proces wydaje się być uruchomiony na węźle gdzie został stworzony. Jest to określane jako Uniqe Home Node. Jeśli jest to moźliwe to proces wykorzystuje lokalne zasoby, jedank czasem musi wykonać wywołania na UHN.

Przeniesiony kontekst użytkownika, nazywany zdalnym, zawiera wszystkie dane dotyczące procesu jak: kod, stos, dane, mapy pamięci i nawet rejestry. W sytuacji gdy zdalny process wymaga wywołań systemowych, OpenMOSIX prechwytuje je i przekierowuje do delegata.

(rysunek)

Głównym narzędziem algorytmu do zarządzania zasobami jest PPM (Pre-emptive Process Migration). Tak długo jak wymagania dla zasobów jak np. CPU są poniżej okresłonego pułapu wszystkie procesy są wykonywane w węźle domowym. Kiedy te wymagania przekroczą określony poziom, niektóre procesy zostają przenoszone do innych węzłów. Decyzja podejmowana jest nie tylko na podstawie obciążenia, lecz także na podstawie wolnej pamięci.

# Równoważenie obciążenia w OpenMOSIX

Głównymi komponentami równoważenia obciążenia w OpenMOSIX są rozprowadzanie ingormacji oraz migracja procesów. Deamon rozprowadzania informacji jest odpoiwedzialny za rozprowadzanie informacji o obciążeniu do innych węzłów klastra. Deamon jest uruchomiony na każdym węźle i jest odpowiedźalny za wysyłanie i odbiór informacji o obciążeniu do/z innych węzłów klastra.

Wysyłanie polega na przekazywaniu inforamcji co sekundę do dwuch losowo wybranych węzłów. Pierwszy węzeł jest wybrany ze wszystkich które były ostatnio podłączone do węzła przekazując swoją informację o obciążeniu. Drugi jest wybierany spośród reszty węzłów klastra.

Deamon pracujący na węźle otrzymującym informację zamienia wartości lokalnego wektora obciążenia. Standardowa implementacja wykorzystuję kolejkę FIFO z ośmioma pozycjami.

# Kerrighed

Kontekst zagadnienia – dlaczego nie wystarcza jedna jednostka

Jak to dziala w systemach oiperacyjnych

Klastry –ssi

P2p/broker

Algorytm oparty na metodzie cen – polegam na informacjach od prowadzącego.

Jakie rozwiązania są stowsowane?

Linux/unix/macOS BSD

Inne systemy klastrowe

Str. 45

Condor

**Load Balancing Clusters**