Clusterproc – zarządzanie procesami w systemach klastrowych

Dokumentacja wstępna do projektu z przedmiotu RSO

Marcin Pawlak
mpawlak1@mion.elka.pw.edu.pl

1 Wstęp....1
2 Inicjalizacja modułu clusterproc....2
3 Przydział numerów identyfikacyjnych procesów (PID`s)....2
4 Zmiany w task structure....3
5 System /proc....4
6 Relacje parent/child....4
7 Relacje miedzy grupami wątków....5

1 Wstęp

Clusterproc jest pakietem zmian, które umożliwiają zarządzanie procesami w systemach klastrowych takich jak OpenSSI. Najważniejsze ze zmian postaram się przedstawić w niniejszej dokumentacji projektowej. Clusterproc został stworzony aby zapewnić:

- unikalne w skali klastra numery identyfikacyjne procesów (PID`s)
- widoczność i możliwość dostępu do procesów z jakiegokolwiek węzła klastra

- rozproszone relacje między procesami, parami parent – child, grupami procesów
- możliwość przemieszczania się wykonywanych procesów miedzy węzłami klastra (np. następna instrukcja wykonywana na innym węźle)(process migration)
- możliwość kontynuacji procesu nawet jeśli węzeł, na którym dany proces został stworzony opuści klaster
- możliwość utrzymania relacji miedzy procesami, niezależnie od tego który węzeł klastra ulegnie awarii
- pełny (i opcjonalny) /proc/<pid> dla wszystkich procesów na wszystkich węzłach klastra
- możliwość wsparcia dla współdzielonego głównego systemu plików lub dla głównego systemu plików dla każdego z węzłów
- wspomaganie do 64000 węzłów w klastrze, z możliwością opcjonalnego wsparcia dla większej ilość węzłów

2 Inicjalizacja modułu clusterproc

Moduł clusterproc jest wgrywany w fazie ramdisc/initramfs lub krótko po niej. Oczekuje się, iż komunikacja międzywęzłowa jest już wczytana i znane są już wartości cluster_maxnodes (maksymalna ilość węzłów w klastrze) oraz wartość cluster_this_node (numer lokalnego węzła). Oczekuje się także, że cluster_initnode oraz cluster_single_init są także ustawione. Cluster_single_init jest flagą, która mówi clusterproc czy dokonać single init dla całego klastra czy dokonać init dla węzłów klastra. Jeśli cluster_single_init jest ustawiona, cluster_initnode mówi nam na którym węźle ma zostać dokonany init.

Przebieg inicjalizacji clusterproc jest następujący:

- dostosowanie pid_max aby odzwierciedlało ile bitów zostaje na unikalny wyróżnik numerów procesów (szerzej omówione w punkcie 3 dokumentacji)
- przydział struktur danych do modułu clusterproc
- dla każdego istniejącego procesu przydziel i uruchom strukturę clusterproc
- rejestracja z usługą cluster membership
- rejestracja z usługą komunikacji międzywęzłowej

3 Przydział numerów identyfikacyjnych procesów (PID`s)

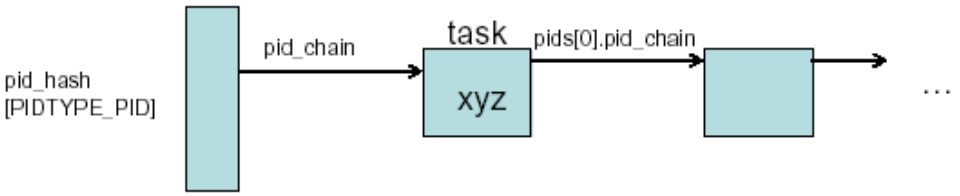
Ważna własnością systemów klastrowych są unikalne w skali klastra numery procesów (process id's) zgodne ze standardowym pid_t. Każdy węzeł klastra posiada własny zakres przydzielanych numerów identyfikacyjnych procesów - może zarządzać przydziałem tych numerów. Z taką strategią przydziału fork może być całkowicie lokalny i śledzenie procesów jest znacznie uproszczone ponieważ przeszukujemy tylko węzeł, który zarządza danym zasięgiem pidów.

Metoda przydziału zakresu pidów dla poszczególnych węzłów wygląda następująco: numer węzła kodowany jest na bitach wysokich a jednoznaczny identyfikator na bitach niskich numeru procesu. Przykładowo jeśli jednoznacznemu identyfikatorowi przydzielimy 16 bitów daje nam to do 65000 unikalnych procesów dla każdego węzła.

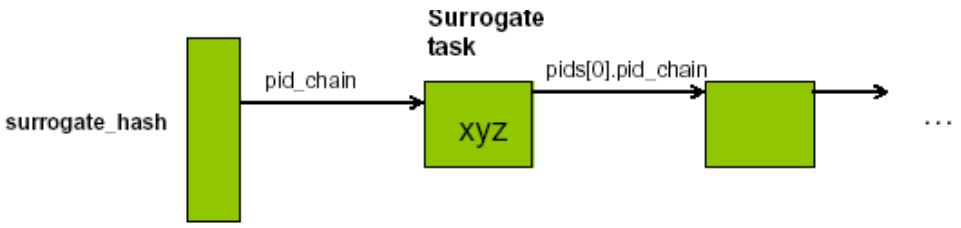
clusterproc_pid_alloc() – wstawia numer węzła do numeru pid następnie zwraca nowy pid, wykorzystuje cluster_maxnodes(zmienna konfiguracyjna określającą maksymalną ilość węzłów w klastrze) - dzięki niej jest wiadome ile bitów jest koniecznych do przydzielenia na numer węzła a co za tym idzie ile bitów zostaje na unikalny identyfikator.

4 Zmiany w task structure

Każdy proces w systemie Linux jest reprezentowany przez strukturę task_struct. Struktura jest dynamicznie alokowana przez jądro w czasie tworzenia procesu. Task_struct jest zdefiniowana w pliku include/linux/sched.h. Zmiany jakie wprowadza rozwiązanie clusterproc polegają na: dodaniu nowego bitu (PF_REMOTE) do flag procesu oraz wskaźnika do struktury clusterproc. PF_REMOTE będzie znajdowała się w task->flags. Flaga ta będzie sygnalizować, że struktura odnosi się do procesu wykonującego się na innym węźle. Niezbędne informacje o takim procesie przechowywane są w strukturze clusterproc. Ponieważ mamy do czynienia z rozproszonymi relacjami miedzy procesami (remote children, remote patent, etc.) tworzone są „zastępcze” task_struct (surrogate task structure) dla procesów nie wykonujących się na danym węźle. Takie struktury nie będą posiadały stosu, nie będą haszowane w tablicach pid_hash[PIDTYPE_PID], nie będą na liście init_task. Będą miały ustawioną flagę PF_REMOTE i nie będą nigdy wykonywalne.



Dla procesu xyz wykonującego się na swoim źródłowym węźle



Struktury dla źródłowego węzła gdy proces wykonuje się na innym węźle

5 System /proc

System plików /proc jest całkowicie wirtualny - nie istnieje na dysku. Jest tworzony i utrzymywany przez jądro w pamięci. Używany jest w celu dostarczenia informacji o systemie (oryginalnie o działających procesach, stąd nazwa).

- readdir będzie przedstawiał wszystkie procesy z wszystkich węzłów oraz inne pliki /proc
- katalogi /proc/node# przekierowujące do katalogów /proc na poszczególnych węzłach
- readdir dla /proc/node# będzie pokazywał tylko procesy wykonujące się na określonym węźle

6 Relacje parent/child

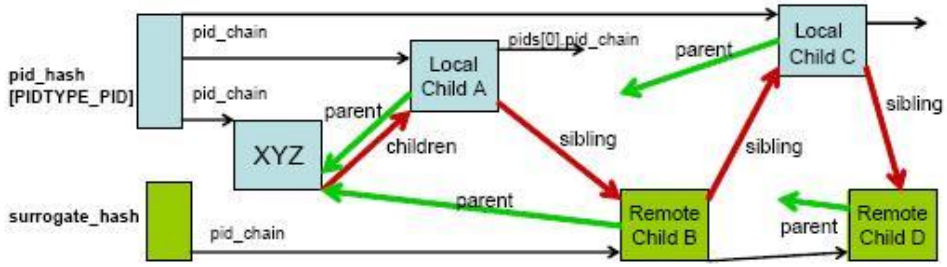
Pełna lista children/sibling jest utrzymywana na węźle, na którym proces macierzysty jest wykonywany.
Jedną z dróg osiągnięcia tego celu jest połączenie tylko lokalnych potomków z procesem macierzystym (proces macierzysty posiadał będzie strukturę uzupełniającą dla procesów potomnym na innych węzłach), w tym modelu operacje sys_wait4()* będą musiały kontaktować się z każdym węzłem na którym procesy potomne są wykonywane i sprawdzać czy istnieje proces, na który trzeba czekać.

*funkcja wait wstrzymuje proces wywołujący do momentu, aż jeden z procesów potomnych przestanie działać. Jeżeli zakończony potomek juz jest, to funkcja wraca natychmiast, implementacją funkcji wait jest funkcja systemowa sys_wait4()

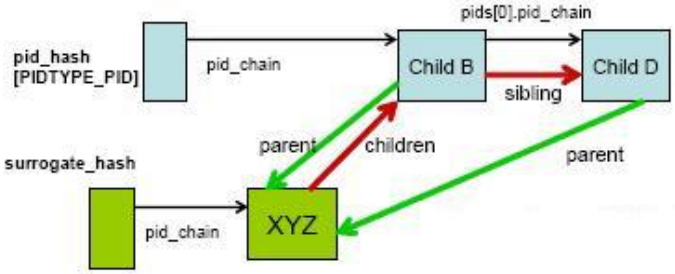
Inna metodą jest posiadanie zastępczej struktury związanej z procesami (surrogate task structure - opisana zostanie ona dokładniej w końcowej wersji dokumentacji projektowej) dla każdego procesu potomnego(nie lokalnego) połączonego z procesem macierzystym.

- na węźle procesu macierzystego
  - połączona lista wszystkich potomków (także nie znajdujących się na lokalnym węźle) (children/sibling list)
  - potomkowie wskazują na proces macierzysty (parent/real_parent pointers)
- na węźle procesu potomnego (rodzic wykonuje się na tym samym węźle) - tak jak wyżej
- na węźle procesu potomnego (rodzic wykonuje się na innym węźle)
  - surrogate task struct dla rodzica

Dla węzła gdzie wykonuje się rodzic xyz:



Dla węzła gdzie wykonuje się proces potomny od xyz:



7 Relacje miedzy grupami wątków

Grupy wątków prawie zawsze dzielą przestrzeń adresową. W konsekwencji tego w rozwiązaniu clusterproc członkowie grupy wątków nie są rozdzielani na inne węzły klastra. Relacje między wątkami pozostają więc podobnie jak pierwotnie w systemie Linux. W obrębie pids[PIDTYPE_TGID] dla liderów grup wątków, wskaźnik pid_list wskazuje na listę członków grupy. Każdy członek wskazuje na lidera grupy polem group_leader w task_struct.