Politechnika Warszawska

Wydział Elektroniki i Technik Informacyjnych
Instytut Automatyki i Informatyki Stosowanej

Działania wstępne dla zadania opracowania nowatorskiego algorytmu energooszczędnego sterowania szybkością pracy serwera

Michał Getka

Raport 17-01

Badanie są wspierane przez Narodowe Centrum Nauki.
Projekt badawczy nr 2015/17/B/ST6/01885.

7 marca 2017

Warszawa 2017

1 Cel dokumentu

Celem dokumentu jest przedstawienie kontekstu i postępu dotychczasowych prac w ramach realizacji działań wstępnych dla zadania czwartego „opracowanie nowatorskiego algorytmu energooszczędnego sterowania szybkością pracy serwera".

W pierwszym rozdziale opisana została motywacja prac realizowanych w ramach zadania. W drugim rozdziale pokrótce opisano kluczowe z punktu widzenia dotychczasowych badań spostrzeżenia dotyczące systemu operacyjnego i procesorowego. Trzeci rozdział stanowi podsumowanie dotychczasowego postępu prac wstępnych.

1.1 Motywacja pracy

Empiryczne prawo Moora, stanowi że ekonomicznie optymalna liczba tranzystorów w układzie scalonym mikroprocesora podwaja się co około 24 miesiące. Wynika to z jednej strony, z rozwoju technologii i optymalizacji procesów technologicznych, z drugiej natomiast z dynamicznie wzrastającego zapotrzebowania na wysoko wydajne systemy obliczeniowe. Nieodłącznie, wraz ze wzrostem mocy obliczeniowej wzrasta pobór energii i w efekcie wydzielane ciepło.

Duże centra danych zużywają porównywalną ilość energii co małe miasto. W ich przypadku, z ekonomicznego punktu widzenia, wskazane jest aby nie tylko rozbudowywać infrastrukturę w celu zwiększania zakresu świadczonych usług, ale również inwestować w rozwiązania pozwalające na zwiększenie wydajności energetycznej systemu. Energooszczędność ma również narastające znaczenie w segmencie konsumenckim. Zasilane bateryjnie urządzenia mobilne stały się nieodłączną częścią codziennego życia. Zmniejszenie zużycia energii pozwala na dłuższy czas pracy pomiędzy cyklami ładowania, towarzyszące mu ograniczenie wydzielania ciepła pozwala natomiast na ograniczenie gabarytów układów chłodzących i w konsekwencji postępującą miniaturyzację urządzeń.

Producenci komponentów systemów komputerowych odpowiadają na zapotrzebowanie rynku proponując urządzenia zawierające mechanizmy zarządzania zużyciem energii. W szczególności układy mikroprocesorowe ostatnich generacji mogą pracować na różnych poziomach wydajności, które to nie muszą ograniczać się tylko do zmiany częstotliwości pracy zegara. W celu zwiększenia efektywności energetycznej przy zachowaniu jakości świadczonych usług, bieżąca wydajność procesora powinna być skorelowana z zapotrzebowaniem na moc obliczeniową niezbędną dla zapewnienie oczekiwanej funkcjonalności systemu w danych warunkach. Oportunistyczne regulatory skalujące częstotliwość procesora, działające w obrębie jądra systemu operacyjnego, dążą do zapewnienia takiej korelacji.

W systemie Linux, dla procesorów opartych o architekturę Intel 64 i IA-32, dostępne są trzy warianty regulatorów oportunistycznych: opracowany przez producenta „Intel P-state", oraz otwarte rozwiązania opracowane przez społeczność programistów „CPUFreq Ondemand" i „CPUFreq Conservative". Mechanizmy te, jakkolwiek prawidłowo realizujące swoje zadania w większości przypadków, oparte są na bardzo prostych mechanizmach – dla przykładu, rdzeń regulatora „Intel P-state" stanowi klasyczny regulator PID.

Przewiduje się możliwość zwiększenia skuteczności działania procesów regulacji poprzez zastosowanie bardziej wyrafinowanych algorytmów sterowania. Ponadto, w obrębie możliwych do odczytu z poziomu systemu operacyjnego rejestrów procesora, dostępnych jest szereg informacji potencjalnie użytecznych dla przebiegu procesu regulacji, które to nie są w obecnie stosowanych rozwiązaniach w tym celu wykorzystywane.

1.2 Cel pracy

Celem rozpoczętych prac w ramach zadania „opracowanie nowatorskiego algorytmu energooszczędnego sterowania szybkością pracy serwer" jest implementacja regulatora poboru mocy procesora, bazującego na informacji o faktycznym zużyciu energii przez procesor, dostępnej w ramach interfejsu RAPL (Running Average Power Limit) procesorów opartych na architekturze Intel 64 i IA-32. Planuje się zastosowanie algorytmu rekurencyjnej metody najmniejszych kwadratów (RLS – recursive least squares) dla modelowania i prognozowania poboru mocy. Planuje się również

przebadanie zasadności uwzględniania informacji dotyczących charakteru bieżącego obciążenia w procesie regulacji, tj. komunikacji aplikacji przestrzeni użytkownika z działającym w jadrze regulatorem w celu informowania o parametrach bieżących i planowanych operacji.

Celem prac opisywanych w niniejszym dokumencie jest zapoznanie się z cechami środowiska pracy regulatora, właściwościami procesorów opartych o architekturę Intel 64 i IA-32 w kontekście obiektów regulacji.

2 Sterowania pracą procesora w systemie Linux

Procesory oparte na architekturze Intel 64 i IA-32 oferują szereg interfejsów pozwalających zarówno na ingerencję systemu operacyjnego w sposób ich działania, jak i odczytywanie informacji dotyczących parametrów ich pracy. Jądro systemu operacyjnego Linux, w oparciu o który prowadzone są badania, zawiera mechanizmy zarządzania energią w ramach systemu komputerowego. Sposób ich implementacji narzuca ideologię działania i realizacji autorskich rozwiązań.

W rozdziale pokrótce opisano kluczowe z punktu widzenia dotychczasowych badań spostrzeżenia dotyczące systemu operacyjnego i procesorowego.

2.1 Jądro systemu Linux

Pierwsza wersja kodu źródłowego jądra systemu Linux opublikowana została w Internecie przez Linusa Torvaldsa w roku 1991. Jądro systemu jest objęte Powszechną Licencją Publiczną GNU (GNU GPL), oznacza to że każdy może bezpłatnie pobrać jego kod źródłowy i wprowadzać w nim dowolne modyfikacje. Jedynym wymogiem jest dalsze udostępnianie wersji zmodyfikowanych na tych samych zasadach. Dzięki temu, dzisiejsza postać jądra jest owocem pracy wielu programistów, komunikujących się głownie za pośrednictwem internetowych list dyskusyjnych.

W przypadku kiedy znaczna grupa osób, o różnych nawykach, pracuje nad jednym projektem, może dojść do sytuacji w której fragmenty przygotowane przez każdą z nich rządzą się zupełnie innymi prawami. Funkcjonalnie, będą one realizować jedną spójną ideę, jednak w kwestii metodyki realizacja poszczególnych funkcjonalności składowych jak i zapisu kodu źródłowego, każda będzie niejako „z innego świata". Aby takich sytuacji uniknąć społeczność zdefiniowała spójny, obowiązujący styl programowania. Składają się na niego zarówno wytyczne dotyczące formatowania kodu, sposobu nadawania nazw zmiennych, unikania pewnych konstrukcji językowych jak i tworzenia komentarzy. Sprawia to że zarówno kod który przygotujemy jest czytelny dla innych członków społeczności, jak i zasoby dotychczas zawarte w repozytoriach będą dla nas łatwiej przyswajalny.

Omówiona kwestia spójności sposobu programowania ma również związek z przyjętym w społeczności sposobem dokumentowania źródeł. Przy modyfikowaniu dotychczas istniejących funkcjonalności, bądź wdrażaniu nowych, które nie są w pewien sposób rewolucyjne, generalnie nie narzuca się tworzenia czy aktualizowania dokumentacji jądra. Przyjmuje się że kod powinien być napisany w sposób na tyle przejrzysty aby sam dla siebie stanowił dokumentację. Pewne wybrane moduły czy aspekty funkcjonowania są co prawda opisane w odpowiednich plikach dokumentacyjnych. Ogranicza się to jednak tylko do wysokopoziomowych funkcjonalności i sposobu ich obsługi, nie natomiast do szczegółowego opisu każdego aspektu implementacji. Za część dokumentacji należy również uznać zapisy dyskusji przeprowadzonych w ramach list dyskusyjnych, na drodze których rzesza autorów skonkludowała ostateczną postać realizacji danej funkcjonalności.

W subiektywnej opinii autora pracy, przyjęte w ramach społeczności standardy mają swoje wady i zalety. Bezdyskusyjnie takie podejście nie zwiększa nakładów pracy, poprzez wymóg tworzenia specjalnej dokumentacji, kładzie to też wyjątkowy nacisk na przestrzeganie przyjętego stylu kodowania, co sprzyja czytelności źródeł. Jako wadę wskazuje się fakt że w celu zrozumienia sposobu realizacji, często najprostszych funkcjonalności, niezbędne jest zapoznanie się z fragmentami kodu zawartymi w wielu funkcjach i definicjach, często na przestrzeni wielu plików źródłowych, co bywa uciążliwe. Niewątpliwie, w posługiwaniu się źródłami jądra systemu Linux w charakterze dokumentacji należy nabrać wprawy.

2.2 Mechanizm zarządzania mocą procesora w systemie Linux

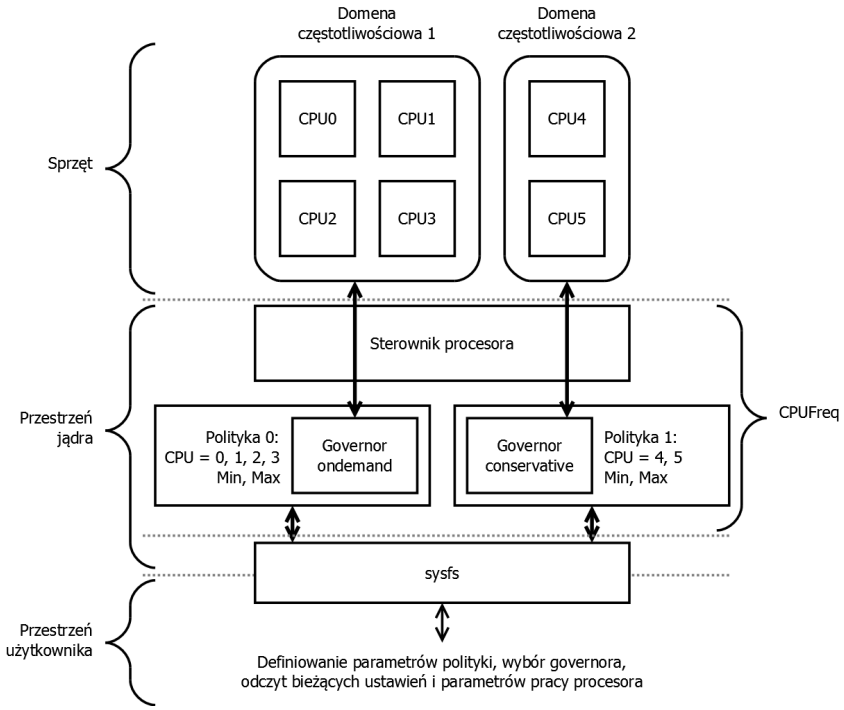
Mechanizmem odpowiedzialnym za zarządzanie mocą procesora w systemie Linux jest CPUFreq, przy czym stwierdzenie o „zarządzaniu" jest tutaj pewnym uszczegółowieniem. W rzeczywistości bowiem CPUFreq, jako taki, nie jest związany z konkretną architekturą procesora. Wspiera on zarówno mikroprocesory klasy x86, jak i inne, takie jak na przykład układy z rdzeniem ARM. W ogólności procesory miewają wbudowane mechanizmy skalowania częstotliwości zegara w odpowiedzi na obciążenie. W takim przypadku moduł CPUFreq, nie steruje pracą procesora, jedynie definiuje górny i dolny limit częstotliwości. Limity te definiowane są w ramach polityki.

Należy w tym momencie zauważyć że spotykane są układy wielordzeniowe w których rdzenie podzielone są na więcej niż jedną domenę częstotliwościową, tzn. w obrębie jednego procesora, zespoły rdzeni logicznych pracują przy różnych częstotliwościach zegara. Identyczna sytuacja występuje w przypadku systemów wieloprocesorowych – rdzenie logiczne widziane przez system operacyjny pracują w ramach różnych domen częstotliwościowych (w szczególności są to odrębne procesory z pojedynczymi domenami). W takich przypadkach dla każdej z domen definiowana jest odrębna polityka.

W przypadku procesorów w których autonomiczne mechanizmy skalowania nie są dostępne lub zdecydowano się je wyłączyć rola CPUFreq jest znacznie większa. Dla polityki każdej z domen częstotliwościowych przypisywany jest algorytm regulatora skalującego częstotliwość. Algorytmy te nazywane są governorami. W systemie Linux dostępnych jest pięć governorów niezależnych od architektury sprzętowej. Trzy z nich mają charakter statyczny – ustawiają częstotliwość na maksymalną (performance), minimalną (powersave) lub stałą, zdefiniowaną przez użytkownika (userspace), dwa pozostałe ustalają częstotliwość w zależności od obciążenia (ondemand i conservative).

Ostatnim komponentem składowym CPUFreq są sterowniki procesora. Są to moduły, dedykowane konkretnym architekturom sprzętowym, pozwalającym na komunikację z konkretnym procesorem w ramach zdefiniowanego, niezależnego od architektury API.

Przykładowa konfiguracja mechanizmu CPUFreq dla systemu wielordzeniowego o dwóch domenach częstotliwościowych bez autonomicznego sterowania częstotliwością przez układ sprzętowy, przedstawiona została na rysunku 1.



Rysunek 1: Przykładowa konfiguracja mechanizmu CPUfreq dla systemu wielordzeniowego o dwóch domenach częstotliwościowych bez autonomicznego sterowania częstotliwością przez układ sprzętowy. Symbole CPUx oznaczają procesory logiczne.
Źródło: opracowanie własne.

2.3 Architektury Intel 64 i IA-32

W procesorach opartych o architekturę Intel 64 i IA-32 w celu zarządzania zużyciem energii nie wpływa się bezpośrednio na częstotliwość pracy zegara. Modyfikowanym parametrem są poziomy wydajności domeny zwane p-state. Definiują one zarówno częstotliwość jak i napięcie zasilania odpowiednich rdzeni.

Co istotne, w ramach rozwiązania sprzętowego zaimplementowanych jest szereg mechanizmów wpływających na bieżący p-state. Są to na przykład mechanizmy autonomicznej, sprzętowej kontroli wydajności (HWP), zabezpieczenia termiczne czy definiowane przez oprogramowanie (system operacyjny) ograniczenie mocy. Uwagę warto poświęcić ostatniemu z wymienionych – RAPL (Running Average Power Limit). Pozwala on na zdefiniowanie maksymalnej mocy dla wybranej domeny. Sygnał mocy zużywanej przez domenę jest filtrowany (filtr uśredniający, o definiowanej przez oprogramowanie szerokości okna) i następnie porównywany ze zdefiniowanym limitem. Jednak z punktu widzenia niniejszej pracy, nie sama funkcjonalność jest istotna. Wśród interfejsów mechanizmu RAPL szczególnie interesujący jest rejestr MSR_PKG_ENERGY_STATUS. Jest to rejestr tylko do odczytu, zawierający informację o faktycznym zużyciu energii przez odpowiednią domenę, która może być przeliczona do jednostek układu SI. Rejestr aktualizowany jest co około 1ms i przepełnia się w czasie rzędu jednej minuty, zależnie od zużycia energii.

W przypadku gdy wspomniany wcześniej mechanizm sprzętowej kontroli wydajności jest nieaktywny lub niedostępny, system operacyjny może wpływać na p-state w obrębie domeny za pośrednictwem rejestrów IA32_PERF_CTL. Co istotne każdy rdzeń logiczny posiada swój rejestr IA32_PERF_CTL. Pomimo tego że fizycznie rdzenie będące w tej samej domenie muszą pracować z tym samym p-state, zawartość rejestru każdego z nich może być różna. Związane to jest z faktem że rolą rejestru IA32_PERF_CTL jest definiowania żądania określonego poziomu wydajności z punktu widzenia operacji przeprowadzanych na danym rdzeniu. W obrębie domeny następuje

przejście do p-state o najniższej (najwyższej wydajności obliczeniowej) żądanej wartości.

2.3.1 Architektura a CPUFreq

Jak zostało wcześniej opisane, pomimo powiązań pomiędzy rdzeniami logicznymi poprzez wspólną domenę częstotliwościową, każdy z nich dysponuje niezależnym rejestrem wpływającym na poziom wydajności p-state. Ponadto zgłaszane mogą być wyłącznie żądania oczekiwanych p-state i nie ma pewności że faktycznie ta wartość zostanie zastosowana, nawet pomijając wpływ mechanizmów zabezpieczeń termicznych.

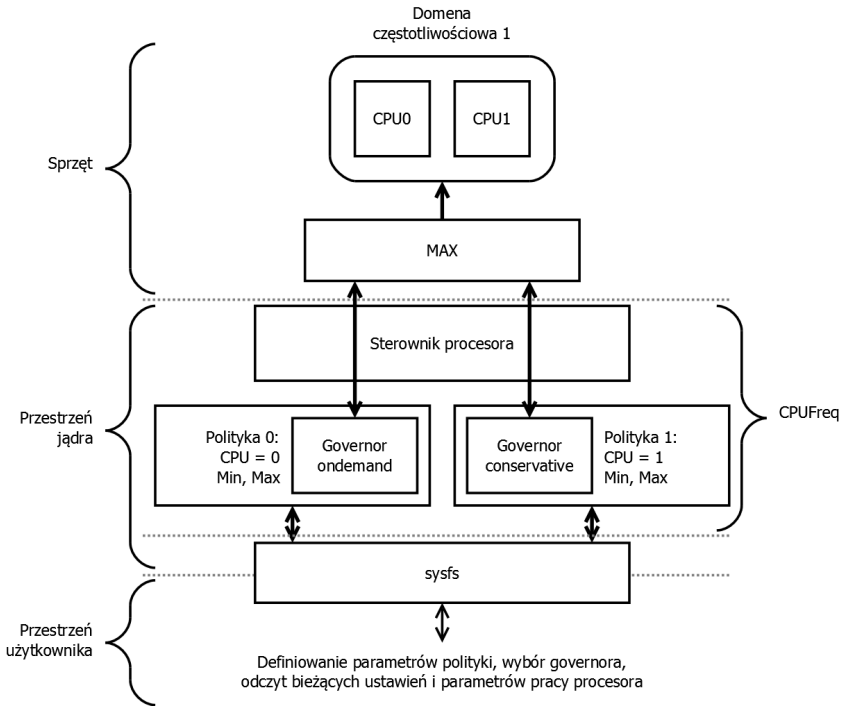
Zdefiniowany w ten sposób interfejs zarządzania wydajnością procesora nie jest w pełni spójny z ideologią działania mechanizmów CPUFreq opisaną w podrozdziale 2.2. Z tego powodu od wersji jądra 3.9 wprowadzone zostały zmiany w sposobie obsługi procesorów architektury Intel 64 i IA-32, które naruszają w pewnym stopniu przeznaczenie komponentów mechanizmów CPUFreq, zapewniając jednak obsługę procesora w sposób przewidziany przez producenta.

Niezależnie od zasięgu domen częstotliwości, CPUFreq postrzega każdy procesor logiczny jako niezależną domenę. Powoduje to że dla każdego z nich wyznaczana jest niezależna wartość estymatora obciążenia a następnie wyznaczana jest nowa wartość sterująca. Każdy z governorów niezależnie, za pośrednictwem sterownika procesora ustawia dla obsługiwanego rdzenia logicznego nową częstotliwość pracy. Pierwsza rozbieżność z ideą CPUFreq, polega na tym że wartości zgłaszane przez governory nie są faktycznie interpretowane jako częstotliwość, a identyfikatory p-state. Druga rozbieżność, objawia się tym że wywołując funkcję API sterownika odpowiedzialną za definiowanie nowych parametrów pracy domeny, faktycznie zgłaszane są żądania wydajności, spośród których wprowadzany p-state wybierany jest już w procesorze.

Podsumowując, w przypadku obsługi procesorów opartych na architekturze Intel 64 lub IA-32, działanie CPUFreq wiąże się z następującymi odstępstwami od pierwotnej ideologii:

- Dla każdego z procesorów logicznych, niezależnie od zasięgu domeny, tworzona jest niezależna polityka obsługiwana przez niezależną instancję governora.
- Wypracowywane przez regulator wartości sterujące są interpretowane jako identyfikatory p-state, nie natomiast jako wartości częstotliwości.
- Wyznaczona przez regulator wartość sterująca nie musi faktycznie zostać zastosowana, regulator nie dysponuje również faktyczną wartością p-state przy jakiej pracuje obsługiwany przez niego procesor logiczny.

Przykładowa konfiguracja mechanizmu CPUFreq dla systemu wielordzeniowego opartego o architekturę Intel 64 lub IA-32 bez autonomicznego sterowania częstotliwością przez układ sprzętowy, przedstawiona została na rysunku 2.



Rysunek 2: Przykładowa konfiguracja mechanizmu CPUfreq dla systemu wielordzeniowego opartego o architekturę Intel 64 lub IA-32 bez autonomicznego sterowania częstotliwością przez układ sprzętowy. Symbole CPUx oznaczają procesory logiczne. Blok MAX oznacza wybór wartości p-state odpowiadającej najwyższemu poziomowi wydajności.
Źródło: opracowanie własne.

3 Postęp prac

W ramach rozpoczętych działań wstępnych zadania zrealizowane zostały następujące aktywności:

- **Zapoznanie się z artykułami naukowymi poruszającymi tematykę energooszczędnego zarządzania energią w systemach komputerowych.**
- **Przygotowanie środowiska badawczego.**
  Na komputerze osobistym wyposażonym w procesor Intel Core i5-3230M zainstalowano system Linux. Dla systemu skompilowano jądro w konfiguracji pozwalającej na swobodę modyfikowania modułów obsługi procesora.
- **Zapoznanie się z regułami implementacji modułów jądra systemu Linux.**
  W oparciu o źródła książkowe, zawartość list dyskusyjnych oraz dokumentacji utworzonej przez środowisko programistów systemu Linux, zebrano niezbędną wiedzę potrzebną do implementacji własnych funkcjonalności w ramach jądra systemu.
- **Zapoznanie się z wybranymi fragmentami dokumentacji procesorów Intel opartych o architekturę Intel 64 i IA-32.**
  Z dokumentu wyodrębnione zostały informacje dotyczące obecnych w procesorach mechanizmów zarządzania energią oraz interfejsów pozwalających modyfikować ich zachowania z poziomu oprogramowania, w szczególności z poziomu jądra systemu operacyjnego. Zapoznano się również ze znaczeniem i sposobem interpretacji szeregu aktualizowanych przez sprzęt rejestrów, zawierających informację zwrotną dla systemu operacyjnego, dotyczącą wydajności i parametrów pracy procesora.

Na najbliższy czas planowane są następujące działania:

- **Wyczerpujące podsumowanie informacji nt. kooperacji procesora i systemu operacyjnego w ramach zadnia sterowania wydajnością systemów wielordzeniowych.** Procesory oparte na architekturze Intel 64 i IA-32 cechują się złożonymi mechanizmami zarządzania energią dla jednostek wielordzeniowych. We wcześniej opracowanych architekturach, system operacyjny miał faktyczną kontrolę nad częstotliwością zegara taktującego pracę zespołu rdzeni. W omawianej architekturze, podejście zmienione zostało na definiowanie przez system żądań wydajności niezależnie dla każdego rdzenia, na podstawie których procesor definiuje odpowiednie parametry swojej pracy. Zmiana ta pociągnęła za sobą również zmiany w działaniu odpowiednich komponentów jądra systemu Linux, które to jednak nie zostały dotychczas udokumentowane w sposób bardziej przejrzysty niż zapisy list dyskusyjnych.