Politechnika Warszawska

Wydział Elektroniki i Technik Informacyjnych
Instytut Automatyki i Informatyki Stosowanej

Przystosowanie jądra systemu Linux do celów zadania opracowania nowatorskiego algorytmu energooszczędnego sterowania szybkością pracy serwera

Michał Getka

Raport 17-03

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

27 września 2017

Warszawa 2017

1 Cel dokumentu

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

W pierwszym rozdziale opisano cel omawianych w ramach raportu prac. W drugim rozdziale zawarto zwięzłe omówienie kluczowych aspektów kooperacji procesora i systemu operacyjnego w ramach zadnia sterowania wydajnością systemów wielordzeniowych zgodnych ze specyfikacją ACPI w systemie Linux. W ramach pewnych konfiguracji sprzętowych, występują aspekty wspomnianej kooperacji, utrudniające lub wręcz uniemożliwiające implementację mechanizmów regulacji adaptacyjnej w ramach istniejących struktur zarządzania energią systemu Linux. W rozdziale trzecim omówiono modyfikację jądra systemu Linux mającą znieść wspomniane ograniczenia.

1.1 Cel pracy

Celem rozpoczętych prac w ramach zadania „opracowanie nowatorskiego algorytmu energooszczędnego sterowania szybkością pracy serwer" jest implementacja regulatora poboru mocy procesora. 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.

Pozyskane i ustrukturyzowane zostały informacji na temat sterowania wydajnością procesora zgodnego z ACPI w systemie Linux. Z przeprowadzonych badań wynika że w pewnych konfiguracjach sprzętowych, mechanizmy kooperacji mechanizmów ACPI i systemu Linux utrudniają lub wręcz uniemożliwiają implementację mechanizmów regulacji adaptacyjnej. Przedstawiony zostały charakter narzucanych ograniczeń oraz modyfikacją jądra systemu pozwalająca ich uniknąć.

2 Sterowanie wydajnością procesora zgodnego z ACPI w systemie Linux - podsumowanie

W raporcie 17-01 "działania wstępne dla zadania opracowania nowatorskiego algorytmu energooszczędnego sterowania szybkością pracy serwera" przedstawiono wstępne rozpoznanie mechanizmów sterowania poziomami wydajności procesora w systemie Linux w kontekście procesorów opartych o architekturę Intel 64 i IA-32. Zadeklarowano również opracowanie wyczerpującego podsumowanie informacji nt. kooperacji procesora i systemu operacyjnego w ramach zadania sterowania wydajnością systemów wielordzeniowych. W niniejszym rozdziale przedstawiono kluczowe wnioski wynikające z rozpoznania wspomnianych zagadnień oraz ich implikacje dla projektu nowatorskiego algorytmu energooszczędnego sterowania szybkością pracy serwera.

2.1 Specyfikacja ACPI

ACPI (*Advanced Configuration and Power Interface*) jest otwartym standardem definiującym model obsługi poszczególnych komponentów systemu komputerowego. Jako obsługę rozumie się ich inicjalizację oraz pośrednictwo w komunikacji systemu operacyjnego z wbudowanymi mechanizmami zarządzania energią. Co istotne, w ramach standardu ACPI definiuje się nie tylko interfejsy obsługi wspomnianych mechanizmów, ale również określa się pewne aspekty specyfikacji ich działania.

Specyfikacja ACPI definiuje pulę stanów, w których znajdować się może system komputerowy i jego komponenty. Stany mają charakter hierarchiczny.

Wyróżniono cztery stany globalne definiowane dla całego urządzenia: uruchomione (G0), uśpione (G1) oraz wyłączone (G2 i G3). W systemie który jest uruchomiony (jest w stanie G0) procesor

może być w jednym ze stanów C wiązanych z mechanizmem usypiania. Stan C0 jest stanem normalnej pracy, stany C1, C2 i dalsze odpowiadają coraz głębszemu uśpieniu. W czasie normalnej pracy procesor może działać z różną wydajnością - różnym jej poziomom odpowiadają stany P, gdzie stan P0 odpowiada najwyższej wydajności. Równolegle do stanów P występują również stany T związane z mechanizmem *throttlingu* używanego do ograniczania wydajności przez mechanizmy ochrony termicznej.

W przypadku systemów wielordzeniowych z reguły występują zależności pomiędzy wydajnością procesorów logicznych. Poszczególne rdzenie korzystają ze wspólnych linii zasilania. Implikuje to, że muszą one pracować z tym samym poziomem wydajności - tym samym stanem P. W szczególności ma to miejsce w urządzeniach wspierających technologię *hyper-threading*, kiedy to dwa procesory logiczne widziane przez system operacyjny są w rzeczywistości jednym fizycznym rdzeniem. Procesory logiczne, które muszą pracować na tym samym poziomie wydajności, korzystają ze wspólnej domeny częstotliwościowej.

Bez względu na to czy istnieją zależności pomiędzy procesorami logicznymi każdy z nich dysponuje swoim rejestrem modyfikującym poziom wydajności. Potencjalnie może to doprowadzić do sytuacji kiedy zawartość rejestrów procesorów z jednej domeny częstotliwościowej nie jest spójna. Takie sytuacje obsługiwane są przez odpowiednie mechanizmy koordynacji. Standard ACPI pozostawia producentom w tej kwestii pewną swobodę. Urządzenie może zawierać wewnętrzny mechanizm koordynacji lub oczekiwać odpowiedniego zarządzania od systemu operacyjnego. Informacja o tym, który z wymienionych wariantów jest realizowany w ramach danego urządzenia udostępniana jest za pośrednictwem zdefiniowanego przez standard ACPI interfejsu. Możliwe warianty to:

- `HW_ALL` - procesor ma wbudowane mechanizmy koordynacji stanów wydajności w obrębie domeny.
- `SW_ALL` - wymagana jest koordynacja na poziomie systemu operacyjnego, nowa wartość stanu wydajności dotycząca domeny częstotliwościowej musi być wprowadzona do odpowiedniego rejestru wszystkich procesorów logicznych w ramach tej domeny.
- `SW_ANY` - wymagana jest koordynacja na poziomie systemu operacyjnego, nowa wartość stanu wydajności dotycząca domeny częstotliwościowej wprowadzana jest do odpowiedniego rejestru dowolnego z procesorów logicznych w ramach tej domeny.

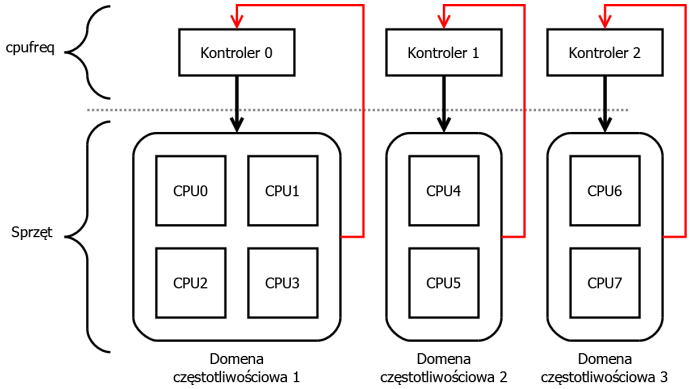
W przypadku wbudowanej koordynacji sprzętowej standard ACPI nie definiuje żadnych wytycznych dla sposobu realizacji tego mechanizmu. Nie istnieje również żaden obiekt ACPI, zawierający informację o sposobie wyboru ostatecznie wprowadzanego stanu P. W praktyce stosuje się mechanizm głosowania. Zawartość odpowiedniego rejestru każdego z procesorów logicznych dzielących wspólną domenę częstotliwościową traktowana jest jako żądanie odpowiedniego poziomu wydajności. Ostatecznie w obrębie domeny częstotliwościowej wprowadzany jest stan P odpowiadający najwyższej z żądanych wydajności. Stosowane są również różne podejścia w kwestii prawa głosu procesorów, które są w różnych stanach uśpienia C.

2.2 Koordynacja stanów wydajności w systemie Linux

W zależności od przyjętych przez producenta procesora założeń dotyczących koordynacji stanów wydajności, moduł *cpufreq*, odpowiedzialny za obsługę mechanizmów sterowania poziomami wydajności procesora w systemie Linux, realizuje swoje zadania stosownie do wymaganego scenariusza.

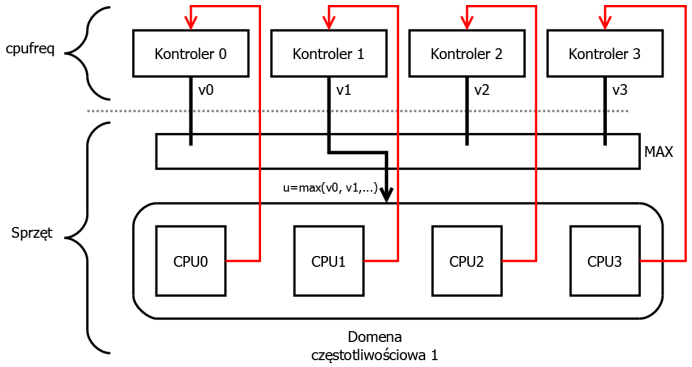
Informacja o zależnościach jest wykorzystywana wyłącznie wtedy, kiedy wymagana jest koordynacja na poziomie systemu operacyjnego (`SW_ALL` lub `SW_ANY`). Każda z domen częstotliwościowych jest wtedy obsługiwana przez jedną politykę, a co za tym idzie jeden kontroler (w szczególności, kontroler może realizować zadania regulacji automatycznej). Zostało to zwizualizowane na rysunku 1. Kolorem czarnym oznaczono sygnał sterujący – żądanie wydajności w ramach domeny częstotliwościowej. Kolorem czerwonym oznaczono sygnał sprzężenia zwrotnego. Na potrzeby prowadzonej

dyskusji, wystarczające jest zdefiniowanie sygnału sprzężenia zwrotnego jako pewnej miary efektywności wykorzystania zasobów sprzętowych procesora przy bieżącym obciążeniu przez aplikacje użytkownika. Każdy z kontrolerów, w ramach sprzężenia zwrotnego, dysponuje odpowiedzią obsługiwanej grupy procesorów logicznych na wyznaczoną przez siebie wartość sygnału sterującego.



Rysunek 1: Przykładowa konfiguracja mechanizmu `cpufreq` dla systemu wielordzeniowego o trzech domenach częstotliwościowych w przypadku realizacji koordynacji poziomów wydajności w wariancie `SW_ALL` lub `SW_ANY`. Kolor czarny – sygnał sterujący, kolor czerwony – sprzężenie zwrotne.
Źródło: opracowanie własne.

Kiedy procesor implementuje wbudowane mechanizmy koordynacji (`HW_ALL`) informacja o zależnościach jest ignorowana - każdy procesor logiczny traktowany jest jak niezależna domena częstotliwościowa, obsługiwana przez niezależny kontroler. Zostało to zwizualizowane na rysunku 2.



Rysunek 2: Przykładowa konfiguracja mechanizmu `cpufreq` dla systemu wielordzeniowego o jednej domenie częstotliwościowej w przypadku realizacji koordynacji poziomów wydajności w wariancie `HW_ALL`. Kolor czarny – sygnał sterujący, kolor czerwony – sprzężenie zwrotne.
Źródło: opracowanie własne.

Na każdy rdzeń logiczny w ramach domeny częstotliwościowej przypada jeden kontroler. Ostatecznie dla całej domeny częstotliwościowej zastosowany zostaje sygnał sterujący odpowiadający najwyższej wydajności, pozostałe sygnały zostają zignorowane. Każdy z kontrolerów w ramach sprzężenia zwrotnego obserwuje wpływ zastosowanej wartości sygnału sterującego na efektywności wykorzystania zasobów obsługiwanego rdzenia logicznego. Nie dysponują one jednak wartościami żądań poziomów wydajności wyznaczonymi przez pozostałe kontrolery. Nie jest zatem możliwe określenie przy jakim rzeczywistym poziomie wydajności uzyskano obserwowany poziom efektywności wykorzystania zasobów. Podsumowując kontrolery obserwują odpowiedź obiektu sterowania na nieznany sygnał sterujący.

3 Wymuszanie metody koordynacji wydajności

Jak zostało już stwierdzone w podrozdziale 2.2, w przypadku kiedy procesor zgłasza systemowi operacyjnemu istnienie wbudowanych mechanizmów koordynacji `HW_ALL`, moduł `cpufreq` ignoruje informacje na temat zasięgów domen częstotliwościowych wiążących poszczególne procesory logiczne. Dostrzeżono że utrata informacji o zasięgu domeny częstotliwościowej uniemożliwia poprawną implementacji mechanizmów identyfikacji parametrów modelu obiektu na potrzeby regulacji adaptacyjnej. Zdecydowano się zmodyfikować bibliotekę `processor_perflib` jądra systemu Linux, pośredniczącą w przekazywaniu informacji nt. oczekiwanego wariantu koordynacji poziomów wydajności od warstwy sprzętowej do modułu `cpuferq`, w taki sposób aby możliwe było wymuszenie działania w wybranym trybie koordynacji bez względu na zgłaszane przez ACPI zalecane warunki pracy.

Moduł `cpufreq` może korzystać z różnych wariantów sterownika pośredniczącego w komunikacji z procesorem. Część z dostępnych sterowników dedykowana jest produktom konkretnych producentów. Są to sterowniki takie jak `intel_pstate` firmy Intel, `powernv-cpufreq` dla procesorów IBM POWER, seria sterowników `powernow` dla rozwiązań firmy AMD, czy `pcc-cpufreq` wykorzystujący interfejs *Processor Clocking Control* dostępny w niektórych urządzeniach serwerowych produkowanych przez HP. W przypadku tych sterowników, dedykowanych ograniczonej puli urządzeń, oczekiwany wariant koordynacji był znany już na etapie implementacji. Moduły te nie odnoszą się do informacji nt. oczekiwanego wariantu koordynacji stanów wydajności procesora, zgłaszanego za pośrednictwem zdefiniowanego przez standard ACPI interfejsu – odpowiednie rozwiązania są statycznie zrealizowane w kodzie sterowników. W takich przypadkach nie istnieje łatwa metoda modyfikacji wariantu koordynacji stanów wydajności. Generycznym sterownikiem jest `acpi-cpufreq` - procesory wszystkich producentów, zgodne ze standardem ACPI mogą być obsługiwane za jego pośrednictwem. W szczególności obsługuje on wszystkie trzy warianty realizacji mechanizmów koordynacji poziomów wydajności. Tylko w przypadku pracy modułu `cpufreq` ze sterownikiem `acpi-cpufreq` podjęte działania, mające na celu umożliwienie narzucania wariantu obsługi koordynacji poziomów wydajności są zasadne.

3.1 Tworzenie polityki[1]

W celu wprowadzenia zmian w sposób odpowiedni prześledzono sposób rejestracji procesorów logicznych w module `cpufreq` pracującym ze sterownikiem `acpi-cpufreq` oraz mechanizm tworzenia polityk. Przy uruchamianiu systemu, następuje inicjalizacja modułu `cpufreq` w tym inicjalizacja sterownika procesora. Jedną z wykonywanych czynności jest pobranie informacji o strukturze domen częstotliwościowych z obiektów `_PSD` każdego z procesorów. Obiekt `_PSD` stanowi zdefiniowany przez standard ACPI interfejs dostarczający systemowi operacyjnemu informacji nt. procesorów logicznych systemu komputerowego.

Listing 1: Inicjalizacja procesora w `processor_perflib`.

```
int acpi_processor_preregister_performance(...)
{
 // [...]
 /* Call _PSD for all CPUs */
 for_each_possible_cpu(i) {
  pr = per_cpu(processors, i);
  // [...]
  // Wczytanie informacji _PSD każdego z procesorów do struktur pr
  if (acpi_processor_get_psd(pr)) {
   retval = -EINVAL;
   continue;
  }
 }
```

[1]Opis działania mechanizmów przedstawionych w niniejszym rozdziale opracowany został na podstawie źródeł jądra systemu Linux w wersji 4.11.12

```
 // [...]
 /*
 * Now that we have _PSD data from all CPUs, lets setup P-state
 * domain info.
 */
 for_each_possible_cpu(i) {
  pr = per_cpu(processors, i);
  // [...]
  pdomain = &(pr->performance->domain_info);
  cpumask_set_cpu(i, pr->performance->shared_cpu_map);
  // [...]
  // Mapowanie żądanego typu koordynacji do odpowiedniego
  // wariantu pracy cpufreq
  if (pdomain->coord_type == DOMAIN_COORD_TYPE_SW_ALL)
   pr->performance->shared_type = CPUFREQ_SHARED_TYPE_ALL;
  else if (pdomain->coord_type == DOMAIN_COORD_TYPE_HW_ALL)
   pr->performance->shared_type = CPUFREQ_SHARED_TYPE_HW;
  else if (pdomain->coord_type == DOMAIN_COORD_TYPE_SW_ANY)
   pr->performance->shared_type = CPUFREQ_SHARED_TYPE_ANY;
  // Budowanie maski procesorów dzielących tą samą domenę
  // częstotliwościową
  for_each_possible_cpu(j) {
   if (i == j)
    continue;
   // [...]
   match_pdomain = &(match_pr->performance->domain_info);
   if (match_pdomain->domain != pdomain->domain)
    continue;
   /* Here i and j are in the same domain */
   // [...]
   cpumask_set_cpu(j, pr->performance->shared_cpu_map);
  }
  // [...]
 }
 // [...]
}
```

Ostatecznie w pamięci, dla każdego z procesorów zaalokowane zostają struktury danych zawierające informację o oczekiwanym wariancie koordynacji `shared_type` oraz mapę procesorów logicznych dzielących daną domenę częstotliwościową `shared_cpu_map`.

W dalszej kolejności działające procesory są rejestrowane w ramach modułu `cpufreq`. Dla każdego z nich wywołana jest funkcji `cpufreq_online`. W niej sprawdzane jest, czy istnieje już polityka obsługująca dany procesor.

Listing 2: Fragment funkcji `cpufreq_online` – sprawdzanie czy dla procesora istnieje polityka.

```
/* Check if this CPU already has a policy to manage it */
policy = per_cpu(cpufreq_cpu_data, cpu);
if (policy) {
 // [...] Jeśli istniejąca polityka jest aktywna wyjdź z funkcji
 if (!policy_is_inactive(policy))
  return cpufreq_add_policy_cpu(policy, cpu);
 /* This is the only online CPU for the policy.  Start over. */
 // Jeśli jest nieaktywna kontynuuj
 new_policy = false;
 // [...]
} else {
 // Jeśli nie istnieje polityka dla tego procesora utwórz nową
 new_policy = true;
```

```
 policy = cpufreq_policy_alloc(cpu);
 // [...]
}
```

Jeśli polityka nie istnieje lub istniejąca polityka jest nieaktywna – żaden inny procesor polityki nie jest aktywny – następuje inicjalizacja polityki przez sterownik `acpi-cpufreq`. W ramach tej inicjalizacji następuje odwołanie do wyodrębnionych wcześniej informacji o koordynacji i domenach częstotliwościowych.

Listing 3: Inicjalizacja polityki w sterowniku `acpi-cpufreq`.

```
static int acpi_cpufreq_cpu_init(...)
{
 // [...]
 perf = per_cpu_ptr(acpi_perf_data, cpu);
 // [...]
 policy->shared_type = perf->shared_type;
 /*
 * Will let policy->cpus know about dependency only when software
 * coordination is required.
 */
 if (policy->shared_type == CPUFREQ_SHARED_TYPE_ALL ||
  policy->shared_type == CPUFREQ_SHARED_TYPE_ANY) {
  cpumask_copy(policy->cpus, perf->shared_cpu_map);
 }
 // [...]
}
```

Zgodnie z wcześniejszymi stwierdzeniami informacja o strukturze domen częstotliwościowych `shared_cpu_map` jest wykorzystywana wyłącznie kiedy wymagana jest koordynacja na poziomie systemu operacyjnego - tryby `SW_ALL` lub `SW_ANY`.

Nowo utworzona polityka jest następnie wiązana ze wszystkimi procesorami logicznymi które ma ona obsługiwać.

Listing 4: Inicjalizacja polityki w sterowniku `acpi-cpufreq`.

```
if (new_policy) {
 /* related_cpus should at least include policy->cpus. */
 cpumask_copy(policy->related_cpus, policy->cpus);
}
// [...]
if (new_policy) {
 // [...]
 for_each_cpu(j, policy->related_cpus) {
  per_cpu(cpufreq_cpu_data, j) = policy;
 // [...]
 }
}
```

W ten sposób jeśli wymagana jest koordynacja na poziomie systemu operacyjnego, dla każdej z domen częstotliwościowych tworzona jest jedna polityka. Jeśli natomiast w procesorze istnieją wewnętrzne mechanizmy koordynacji dla każdego z procesorów logicznych utworzona zostanie niezależna polityka.

3.2 Implementacja

Pierwszą funkcjonalnością, którą należy wprowadzić, to utworzenie nowego parametru jądra pozwalającego ostatecznemu użytkownikowi zdefiniować, w jaki sposób ma się odbywać koordynacja. Rejestracja urządzeń i tworzenie polityk realizowana jest w czasie uruchamiania systemu, nie

może to zatem być parametr modyfikowany i odczytywany w czasie działania systemu. Odpowiednim rozwiązaniem jest wykorzystanie parametrów linii poleceń jądra modyfikowanych z poziomu programu rozruchowego (*boot loader*). Zdefiniowano parametr `acpi_cpufreq_coord_type_force` przyjmujący jedną z trzech wartości wymuszających odpowiednie tryby koordynacji: `sw_all`, `sw_any`, `hw_all`.

Listing 5: Definicja i obsługa utworzonego parametru modułu.

```
/*
 * This variable contains early init parameter acpi_cpufreq_coord_type_force
 * value.
 */

static unsigned int coord_type_force = CPUFREQ_SHARED_TYPE_NONE;

// [...]

/*
 * Early init parameter acpi_cpufreq_coord_type_force declaration and parsing.
 * This parameter allows to force any of possible ACPI CPU performance
 * level coordination variant.
 *
 * Possible values for early init acpi_cpufreq_coord_type_force parameter:
 *      - sw_all
 *      - sw_any
 *      - hw_all
 */

static int __init coord_type_force_setup(char *str)
{

        if (!str)
                return -EINVAL;

        pr_debug(PREFIX "Param value provided: '%s'", str);

        if (strcmp(str, "sw_all") == 0)
                coord_type_force = CPUFREQ_SHARED_TYPE_ALL;
        else if (strcmp(str, "sw_any") == 0)
                coord_type_force = CPUFREQ_SHARED_TYPE_ANY;
        else if (strcmp(str, "hw_all") == 0)
                coord_type_force = CPUFREQ_SHARED_TYPE_HW;

        return 0;

}

early_param("acpi_cpufreq_coord_type_force", coord_type_force_setup);
```

Oczekiwanym efektem jest zmuszenie modułu `cpufreq` do realizacji koordynacji stanów wydajności w sposób inny niż zdefiniowany dla danego urządzenia. Intencją nie jest natomiast nadpisanie informacji o faktycznie zgłaszanym przez ACPI oczekiwanym wariancie realizacji tej funkcjonalności. Wobec tego właściwym jest zmodyfikowanie parametru `shared_type` związanego z procesorem, pozostawiając w niezmienionej postaci parametr `coord_type` opisujący domenę.

Listing 6: Modyfikacja parametru `shared_type` w ramach inicjalizacji procesora w sterowniku `processor_perflib`.

```
if (coord_type_force == CPUFREQ_SHARED_TYPE_NONE) {
```

```
        if (pdomain->coord_type == DOMAIN_COORD_TYPE_SW_ALL)
                pr->performance->shared_type = CPUFREQ_SHARED_TYPE_ALL;
        else if (pdomain->coord_type == DOMAIN_COORD_TYPE_HW_ALL)
                pr->performance->shared_type = CPUFREQ_SHARED_TYPE_HW;
        else if (pdomain->coord_type == DOMAIN_COORD_TYPE_SW_ANY)
                pr->performance->shared_type = CPUFREQ_SHARED_TYPE_ANY;

} else {

        // Force requested coordination type
        pr->performance->shared_type = coord_type_force;

        switch (coord_type_force) {
                case CPUFREQ_SHARED_TYPE_ALL:
                        pr_info(PREFIX "CPU performance level coordination
type forced to sw_all\n");
                        break;
                case CPUFREQ_SHARED_TYPE_ANY:
                        pr_info(PREFIX "CPU performance level coordination
type forced to sw_any\n");
                        break;
                case CPUFREQ_SHARED_TYPE_HW:
                        pr_info(PREFIX "CPU performance level coordination
type forced to hw_all\n");
                        break;
                default:
                        pr_err(PREFIX "CPU performance level coordination
type forced to unknown (%x)\n", coord_type_force);
        }

}
```

3.3 Prezentacja działania

Prezentację działania rozwiązania stanowią poniższe dwa listingi konsoli. Utworzone zostały one na komputerze osobistym wyposażonym w procesor ze sprzętowym mechanizmem koordynacji stanów wydajności i jedną domeną częstotliwościową. Pierwszy z nich przedstawia konfigurację polityk utworzoną na podstawie informacji ACPI, drugi zawiera konfigurację przy zastosowaniu parametru `acpi_cpufreq_shared_type_force=sw_all`.

Listing 7: Domyślna konfiguracja *boot loadera* i uzyskana struktura polityk.

```
[getka@*** ~]$ cat /proc/cmdline
BOOT_IMAGE=/vmlinuz-4.11.12-gtk root=/dev/mapper/fedora-root ro
rd.lvm.lv=fedora/root rd.lvm.lv=fedora/swap rhgb quiet intel_pstate=disable
[getka@*** ~]$ cat /sys/devices/system/cpu/cpufreq/policy0/freqdomain_cpus
0 1 2 3
[getka@*** ~]$ more /sys/devices/system/cpu/cpufreq/policy*/related_cpus
::::::::::::::
/sys/devices/system/cpu/cpufreq/policy0/related_cpus
::::::::::::::
0
::::::::::::::
/sys/devices/system/cpu/cpufreq/policy1/related_cpus
::::::::::::::
1
::::::::::::::
/sys/devices/system/cpu/cpufreq/policy2/related_cpus
```

```
::::::::::::::
2
::::::::::::::
/sys/devices/system/cpu/cpufreq/policy3/related_cpus
::::::::::::::
3
```

Listing 8: Konfiguracja *boot loadera* wymuszająca wariant koordynacji `SW_ALL` i uzyskana struktura polityk.

```
[getka@*** ~]$ cat /proc/cmdline
BOOT_IMAGE=/vmlinuz-4.11.12-gtk root=/dev/mapper/fedora-root ro
rd.lvm.lv=fedora/root rd.lvm.lv=fedora/swap rhgb quiet intel_pstate=disable
acpi_cpufreq_coord_type_force=sw_all
[getka@*** ~]$ cat /sys/devices/system/cpu/cpufreq/policy0/freqdomain_cpus
0 1 2 3
[getka@*** ~]$ more /sys/devices/system/cpu/cpufreq/policy*/related_cpus
0 1 2 3
```

Widoczne jest że w drugim przypadku powstała wyłącznie jedna polityka, związana ze wszystkimi czterema procesorami logicznymi, co jest zgodne ze strukturą domen częstotliwościowych.

3.4 Postęp prac

W ramach działań związanych z przystosowaniem jądra systemu Linux zrealizowane zostały następujące aktywności:

- **Wyczerpujące podsumowanie informacji nt. kooperacji procesora i systemu operacyjnego w ramach zadnia sterowania wydajnością systemów wielordzeniowych.**
  Aktywność zadeklarowana w ramach planowanych zadań opisanych w raporcie 17-01 „Działania wstępne dla zadania opracowania nowatorskiego algorytmu energooszczędnego sterowania szybkością pracy serwera". Zwięzłe omówienie kluczowych aspektów przedstawione zostało w rozdziale 2. niniejszego raportu, szczegółowy opis zawarty zostało w artykule „Aspekty energooszczędnego sterowania wydajnością pracy procesora systemu komputerowego zgodnego z ACPI w systemie Linux", oczekującym na publikację w czasopiśmie *Przegląd Elektrotechniczny*, ISSN 0033-2097.

- **Implementacja modyfikacji jądra systemu Linux, pozwalającej na wymuszanie dowolnego wariantu koordynacji stanów wydajności procesorów zgodnych ze standardem ACPI.**
  W szczególności rozwiązanie umożliwia wymuszenie koordynacji stanów wydajności z poziomu systemu operacyjnego, skutkujące obsługą całych domen częstotliwościowych przez pojedyncze instancje kontrolerów modułu `cpufreq`. Taki wariant działania modułu jest niezbędny dla prawidłowej implementacji mechanizmów regulacji adaptacyjnej.

Na najbliższy czas planowane są następujące działania:

- **Implementacja szablonu kontrolera modułu `cpufreq` na potrzeby rozwoju środowiska uruchomieniowego dla nowatorskiego algorytmu energooszczędnego sterowania szybkością pracy serwera.**
  W ramach zasobów środowiska rozwijającego jądro systemu Linux dostępna jest jedynie szczątkowa dokumentacja modułu `cpufreq` traktująca o zagadnieniach związanych z implementacją własnych wariatów kontrolerów. Niezbędne jest określenie w jaki sposób definiowane są funkcjonalności kontrolerów `cpufreq`, w jaki sposób przebiega ich inicjalizacja i finalizacji, jakie struktury danych biorą udział w tych procesach oraz jakie występują między nimi zależności. W oparciu o taką wiedzę możliwe jest stworzenie ogólnego szablonu modułu kontrolera, oraz – jeśli wystąpi taka konieczność – rozbudowanie go o dodatkowe, niezbędne dla implementacji mechanizmów regulacji adaptacyjnej mechanizmy.