علوم رایانش و فناوری اطلاعات
نشریه علمی انجمن کامپیوتر ایران
مجلد ۱۸، شماره ۱، ۱۳۹۹
صفحات ۴۱- ۳۱
مقاله عادی

زمان‌بندی انرژی وظایف بی‌درنگ موازی اولویت-ثابت در سیستم‌های سایبر-فیزیکی چند هسته‌ای

جمال محمدی[۱*]، مهدی کارگهی[۲]، محمود شیرازی[۳]

*نویسنده مسئول، دریافت: ۹۹/۰۶/۲۳، بازنگری: ۹۹/۰۷/۱۹، پذیرش: ۹۹/۰۹/۱۲

[۱] دانشجوی دکترا، دانشکده مهندسی برق و کامپیوتر، دانشگاه تهران، تهران، ایران
[۲] دانشیار، دانشکده مهندسی برق و کامپیوتر، دانشگاه تهران، تهران، ایران
[۳] استادیار، دانشکده علوم رایانه و فناوری اطلاعات، دانشگاه تحصیلات تکمیلی علوم پایه، زنجان، ایران

چکیده

امروزه با افزایش نیاز محاسباتی سیستم‌های سایبر-فیزیکی، توجه به سیستم‌های چندهسته‌ای افزایش چشمگیر داشته است. نقش وظایف موازی که به‌صورت برنامه‌های چندنخی پیاده‌سازی می‌شوند در بهره‌گیری از امکانات پردازنده‌های چندهسته‌ای و پاسخ به نیازهای روزافزون محاسباتی بسیار پراهمیت است؛ در برخی موارد بدون استفاده از پردازش موازی امکان رعایت موعدهای زمانی وجود ندارد. از سوی دیگر، بسیاری از سیستم‌های سایبر-فیزیکی در مأموریت‌هایی به کار گرفته می‌شوند که آن‌ها را در دریافت انرژی محدود می‌سازد. در این سیستم‌ها باید با مدیریت مناسب انرژی ورودی، وظایف را به نحوی زمان‌بندی کرد که بتوان با توجه به بودجه انرژی تمامی موعدهای زمانی را رعایت نمود. در این مقاله، ابتدا تحلیلی از عدم‌قطعیت مصرف انرژی وظایف موازی ارائه می‌شود و سپس یک روش برای زمان‌بندی وظایف بی‌درنگ موازی اولویت-ثابت در سیستم‌های سایبر-فیزیکی چندهسته‌ای با محدودیت انرژی ارائه می‌گردد. نتایج آزمایش‌ها تأثیر مثبت موازی‌سازی وظایف در زمان‌بندی پذیری را نشان می‌دهد. به‌طوری‌که در الگوریتم ارائه شده با کاهش طول مسیر بحرانی به کمتر از ۴۰ درصد، زمان‌بندی پذیری وظایف به‌صورت چشمگیری افزایش می‌یابد.

کلمات کلیدی: زمان‌بندی، سیستم‌های بی‌درنگ، وظایف موازی، انرژی

۱- مقدمه

توسعه سریع فناوری‌های اطلاعاتی در دهه‌های گذشته منجر به دسترسی گسترده به سکوهای محاسباتی نهفته و نیز قابلیت‌های ارتباطی در تقریباً همه انواع اشیا شده است. سیستم‌های سایبر فیزیکی (CPS) با ادغام هوش سایبری و دنیای فیزیکی امکان پاسخگویی به نیازهای فزاینده محاسباتی را فراهم کرده‌اند. این سیستم‌ها با در اختیار گذاشتن امکانات محاسباتی و ارتباطی کاربردهای مهمی نظیر سیستم‌های نظارتی، سیستم‌های حمل‌ونقل و تجهیزات هوشمند پزشکی را تحت پوشش قرار داده است[۱].

از سوی دیگر یکی از نیازمندی‌های اساسی سیستم‌های نهفته امکان به کار گیری آن‌ها در شرایطی است که عموماً محدودیت انرژی وجود دارد[۲]. بسیاری از این سیستم‌ها از نظر انرژی خودمختار بوده و انرژی خود را از محیط استفاده از یک برداشتگر انرژی دریافت کرده و سپس در یک مخزن میانگیر ذخیره می‌کنند. در برخی از چنین سیستم‌هایی دریافت انرژی با نرخی ثابت است و این محدودیت انرژی می‌تواند منجر به نقض قیود زمانی و شکست در زمان‌بندی وظایف شود[۲].

مسئله زمان‌بندی وظایف با محدودیت انرژی ورودی به سیستم در منابع مختلفی مورد بررسی قرار گرفته است [۲][۳][۴][۵]. یک الگوریتم بهینه برای زمان‌بندی وظایف در سیستم‌های با نرخ انرژی ورودی ثابت، الگوریتم PFP_{ASAP}[۱] [۶] است. در این الگوریتم که برای وظایف با اولویت ثابت طراحی شده است، اگر انرژی کافی برای اجرای حداقل یک واحد زمانی از پر اولویت‌ترین وظیفه وجود داشته باشد، این وظیفه سریعاً به اندازه یک واحد زمانی اجرا می‌شود. در غیر این صورت وظایف باید تا زمان ذخیره شدن انرژی به اندازه یک واحد از اجرای پراولویت‌ترین وظیفه منتظر بمانند. بهینگی این الگوریتم نیز در حالت تک نخی و در شرایطی که میزان مصرف انرژی همه وظایف بالاتر از میزان تأمین انرژی است اثبات شده است[۷]. از آنجایی که در سیستم‌های سایبر-فیزیکی طیف گسترده‌ای از اطلاعات از حسگرها دریافت

می‌شود، سیستم باید قادر به پردازش همگی آن‌ها باشد. وظایف موجود در این سیستم‌ها (نظیر برنامه‌های مرتبط با بینایی کامپیوتری، تشخیص الگو، برنامه‌ریزی حرکتی) دارای نیاز محاسبات بسیار زیاد هستند و این نیاز محاسباتی روزبه‌روز به دلیل استفاده از الگوریتم‌های دقیق‌تر، پرداختن به جزییات بیشتر (به عنوان مثال استفاده از تصاویر با تفکیک‌پذیری بالاتر در بینایی ماشین [۸]) و غیره در حال افزایش است تا جایی که رسیدن به موعدهای زمانی چنین سیستم‌هایی با استفاده از سیستم‌های تک پردازنده‌ای و محاسبات تک نخی امکان‌پذیر نیست. در چنین شرایطی نیاز به موازی‌سازی وظایف و استفاده از پردازنده‌های چند هسته‌ای وجود دارد. در حوزه زمان‌بندی وظایف موازی، یکی از روش‌های ایده آل که فارغ از جزییات چند نخی وظایف را به درستی زمان‌بندی می‌کند، الگوریتم زمان‌بندی فدرال [۹] است. در این روش تنها با داشتن مجموع زمان اجرا، طول مسیر بحرانی و موعد وظیفه، حداقل تعداد هسته‌های لازم برای رسیدن به موعد زمانی محاسبه شده و پس از آن به ازای هر یک از وظایف پردازنده اختصاصی تخصیص داده می‌شود.

در این مقاله به بررسی مسئله زمان‌بندی وظایف موازی اولویت ثابت در سیستم‌هایی با نرخ انرژی ورودی ثابت می‌پردازیم. برای این کار از روش زمان‌بندی فدرال استفاده می‌شود و تعداد هسته‌های اختصاصی هر یک از وظایف به نحوی تعیین می‌شود که امکان جبران تأخیر زمانی ناشی از تأمین انرژی وظایف پراولویت‌تر با موازی‌سازی بیشتر وجود داشته باشد. سپس با استفاده از یک الگوریتم مشابه $\mathrm{PFP_{ASAP}}$ زمان‌بندی آگاه از انرژی انجام می‌شود. برای زمان‌بندی وظایف در این الگوریتم نیاز است که انرژی مورد نیاز یک وظیفه موازی در هر واحد زمانی مشخص شود. با توجه به اینکه در وظایف موازی تا قبل از زمان اجرا اطلاعی در مورد نحوه استفاده از هسته‌ها و در نتیجه نحوه مصرف انرژی وجود ندارد، باید تأمین انرژی نیز با در نظر گرفتن بدترین حالت‌ها انجام شود. به همین منظور در این مقاله تحلیلی از بدترین حالت درخواست انرژی در وظایف موازی نیز ارائه شده است. بر اساس اطلاعات ما این اولین باری است که زمان‌بندی وظایف موازی در سیستم‌های چند هسته‌ای بی‌درنگ با محدودیت انرژی ورودی مورد مطالعه قرار می‌گیرد.

نوآوری‌های این مقاله به‌صورت زیر است:

- ارائه یک تحلیل از بدترین حالت درخواست انرژی در وظایف موازی در الف) حالت عادی، ب) با در نظر گرفتن کران بالای زمان اجرای وظایف موازی.
- ارائه روشی برای محاسبه ظرفیت باتری اختصاصی برای هر وظیفه موازی.
- ارائه روشی برای محاسبه تعداد هسته‌های اختصاصی وظایف موازی با در نظر گرفتن تأخیر دریافت انرژی بر زمان اجرای وظایف.
- ارائه یک الگوریتم زمان‌بندی برای وظایف موازی با توجه به نرخ انرژی ورودی ثابت.

در ادامه ساختار مقاله به صورت زیر سازمان‌دهی شده است: در بخش ۲ مقالات مرتبط با زمان‌بندی وظایف و مدیریت انرژی مورد بررسی قرار گرفته است. در بخش ۳ تو ضیحاتی در مورد پارامترهای مهم وظایف موازی ارائه شده است. مدل سیستم در بخش ۴ معرفی شده است. در بخش ۵ زمان‌بندی فدرال معرفی شده است. در بخش ۶ تحلیل بدترین حالت درخواست انرژی[۲] ارائه شده است. در این بخش ابتدا بدترین حالت درخواست انرژی در وظایف موازی معرفی شده و پس از آن تحلیل دیگری با در نظر گرفتن کران بالای اجرای وظایف موازی معرفی شده است. در بخش ۷ به زمان‌بندی آگاه از انرژی وظایف موازی پرداخته شده است. در این قسمت ابتدا نحوه محاسبه تأخیر تأمین انرژی هر یک از وظایف موازی معرفی می‌شود و پس از آن تعداد هسته هر وظیفه به نحوی تعیین می‌شود که این تأخیر جبران شود. در انتهای این بخش الگوریتم زمان‌بندی ارائه شده است. در بخش ۸ نیز کارایی روش ارائه شده با استفاده از آزمایش‌ها مورد بررسی قرار گرفته و نتایج آن با حالت بهینه مقایسه شده است.

۲- کارهای مرتبط

کارهای مرتبط در حوزه مدیریت انرژی در زمان‌بندی سیستم‌های سایبر-فیزیکی را می‌توان به دو دسته کلی الف) آگاه از انرژی[۳] و ب) بهینه‌سازی انرژی[۴] تقسیم کرد. در حوزه زمان‌بندی آگاه از انرژی هدف رعایت قیود زمانی با توجه به نرخ انرژی ورودی به سیستم است، ولی در حوزه بهینه‌سازی انرژی، هدف حداقل کردن مصرف انرژی است. هدف اصلی از پژوهش جاری زمان‌بندی آگاه از انرژی وظایف موازی است. در ادامه به مهم‌ترین کارهای انجام شده در این حوزه اشاره می‌گردد و پس از آن با توجه به اهمیت کارهای انجام شده در حوزه بهینه‌سازی مصرف انرژی، چند مورد مهم که از مدل وظایف موازی استفاده کرده‌اند نیز مورد توجه قرار می‌گیرند.

اولین پژوهش مهمی که در حوزه زمان‌بندی آگاه از انرژی به مسئله زمان‌بندی در سیستم‌هایی با برداشتگر انرژی[۵] پرداخته است مقاله Allavena و همکاران [۱۰] است. در این تحقیق این مسئله تحت یک مدل مبتنی بر فریم حل شده است که تمام وظایف در آن دارای دوره یکسان هستند. در [۱۱] این مسئله با همین فرضیات بررسی شده است. سپس Moser و همکاران [۱۲] یک الگوریتم بهینه به نام LSA[۶] برای سیستم‌های تک پردازنده‌ای ارائه داده‌اند که در فرضیات آن‌ها امکان تغییر فرکانس پردازنده برای تنظیم بدترین زمان اجرای وظایف (WCET[۷]) در نتیجه تطابق با نرخ انرژی ورودی وجود دارد. البته در [۱۳] نشان داده شده است که این فرضیات واقع بینانه نیست. همچنین در [۱۴] Chetto و همکاران چند الگوریتم غیب‌گو[۸] و چند الگوریتم مکاشفه‌ای برای حل این مسئله در حالت تک پردازنده‌ای ارائه کرده‌اند. ایده اصلی الگوریتم ارائه شده در [۱۴] بر این سیاست استوار است که تا زمانی که سیستم بتواند بدون شکست انرژی عمل کند از یک سیاست نظیر EDF[۹] برای زمان‌بندی استفاده می‌شود و درصورتی‌که شکست انرژی در آینده شناسایی شود سیستم تا جای ممکن و تا پر شدن مخزن ذخیره‌سازی انرژی متوقف می‌شود. همچنین یکی از الگوریتم‌های مطرح شده در [۱۴] الگوریتم $\mathrm{PFP_{ALAP}}$[۱۰] است که اجرای وظایف را تا جای ممکن به تأخیر می‌اندازد. در مقابل در [۱۵] بهینه‌گی الگوریتم $\mathrm{PFP_{ALAP}}$ به چالش کشیده شده و با ارائه مثالی عدم بهینه‌گی آن نشان داده شده است. در این مقاله الگوریتم دیگری به نام $\mathrm{PFP_{ASAP}}$ معرفی شده به‌طوری‌که در این الگوریتم اجرای وظایف به محض وجود انرژی کافی انجام می‌شوند و برعکس $\mathrm{PFP_{ALAP}}$، تا زمانی که که انرژی کافی وجود داشته باشد، اجرای هیچ وظیفه‌ی به تأخیر نمی‌افتد. بهینه‌گی این الگوریتم در [۷] اثبات شده است. در [۱۶] حالتی که مخزن ذخیره‌سازی غیر بهینه است مورد بررسی قرار گرفته و یک الگوریتم بهینه مبتنی بر $\mathrm{PFP_{ASAP}}$ برای زمان‌بندی ارائه شده است. در [۱۷] برای حداکثر کردن کارایی در سیستم‌های $(m.k) - \mathrm{firm}$ با نرخ انرژی متغیر روشی ارائه شده که با سوئیچ کردن بین سطوح مختلف کارایی نسبت به تغییرات نرخ انرژی واکنش نشان می‌دهد. در اینجا نیز سیستم مورد نظر یک سیستم تک هسته‌ای بوده و مدل وظایف آن نیز بر اساس یک سیستم تک‌هسته‌ای در نظر گرفته شده است.

در حوزه بهینه‌سازی مصرف انرژی در سیستم‌های بی‌درنگ با وظایف موازی در [۱۸] زمان‌بندی وظایف موازی بی‌درنگ سخت با هدف حداقل کردن مصرف توان پردازنده بررسی شده و در آن برای به حداقل رساندن انرژی مصرفی کل، تنظیم پویای سرعت هسته‌ها در حین برآوردن تمامی موعدهای زمانی مورد بررسی قرار گرفته است. در تحقیق دیگری، در [۱۹] زمان‌بندی آگاه از انرژی وظایف موازی در سیستم‌های چند هسته‌ای ناهمگن مورد بررسی قرار گرفته و به وظایف اجازه داده شده تا به چندین وظیفه فرعی موازی تقسیم شوند. تعداد وظایف فرعی نیز در حین زمان‌بندی به دست می‌آیند و هدف اصلی آنها نیز به حداقل رساندن مصرف انرژی تحت محدودیت‌های مرتبط با موعد زمانی است. در [۲۰] هم یک الگوریتم دو مرحله‌ای برای زمان‌بندی موثر وظایف موازی در مراکزداده سبز ارائه شده است که از مدل وظایف موازی متفاوتی برای زمان‌بندی در قالب ماشین‌های مجازی استفاده می‌نماید.

بر اساس اطلاعات ما، این مقاله نخستین مقاله‌ای است که زمان‌بندی آگاه از انرژی وظایف موازی در سیستم‌های چند هسته‌ای بی‌درنگ با محدودیت انرژی ورودی را مورد مطالعه قرار می‌دهد. لازم به ذکر است که مسئله زمان‌بندی وظایف موازی بر روی سیستم‌های چند هسته‌ای در کلاس مسائل NP-hard قرار می‌گیرد [۲۱]. برای غلبه بر پیچیدگی مسئله در این مقاله از روش زمان‌بندی فدرال [۹] برای زمان‌بندی وظایف استفاده شده است. به این صورت که حداقل تعداد هسته‌های لازم به‌صورت اختصاصی به هر کدام از وظایف موازی تخصیص داده می‌شود. در نتیجه در این حالت رقابت زمانی[۱۱] بین وظایف وجود ندارد و فقط به زمان‌بندی انرژی وظایف پرداخته شده است. علاوه بر این، هدف از این مقاله، بهینه‌سازی (تعداد هسته‌ها یا مصرف انرژی) نیست و از یک روش مکاشفه‌ای برای زمان‌بندی استفاده شده است. پس از معرفی مدل سیستم، مسئله به‌صورت دقیق و رسمی در بخش ۳- ۴ معرفی خواهد شد.

۳- وظایف موازی

در طول دهه گذشته افزایش کارایی تراشه‌های پردازنده‌ها عمدتاً ناشی از افزایش تعداد هسته‌ها بوده است. این امر منجر به تحقیقات گسترده‌ای در توسعه زبان‌های موازی و سیستم‌های زمان اجرا برای بهره‌برداری از توان پردازشی چند هسته‌ای شده است. این تحقیقات منجر به توسعه زبان‌ها و کتابخانه‌های موازی نظیر Cilk[۲۲]، Intel Cilk Plus[۲۳]، IBM X10[۲۴] و غیره شده‌اند. در این زبان‌ها برنامه‌نویس الگوریتم‌های موازی را از طریق ساختارهای زبانی موازی نظیر spawn، sync، fork، join، یا parallel-for توسعه می‌دهد.

```
// Do some sequential work                          Sequential
foo();
// Do the first parallel segment
 parallel_for (i=1; i <= 2; i++) {                  Parallel
    firstfunc(i);
}
// Other sequential work                            Sequential
bar();
// Do the second parallel segment
 parallel_for (i=1; i <= 3; i++) {                  Parallel
     second_func(i);
}
// The last sequential work                         Sequential
baz();
```

شکل ۱- یک نمونه کد که شامل بخش‌های موازی و سری است.

در ادامه این بخش ابتدا مفهوم قسمت سری و موازی وظایف بیان شده و سپس مفهوم مسیر بحرانی توضیح داده می‌شود. این مفاهیم در تحلیل زمان اجرا و نیز تعیین تعداد هسته‌های هر وظیفه استفاده خواهند شد.

۳-۱- بخش سری و موازی وظایف

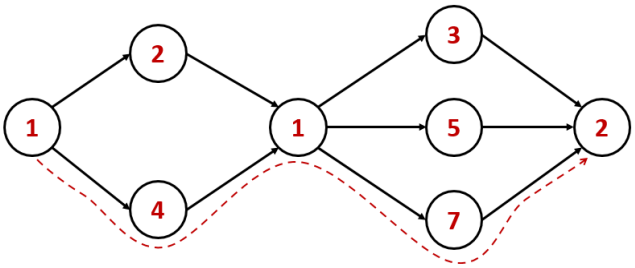
برنامه‌های توسعه یافته با زبان‌ها و کتابخانه‌های موازی شامل بخش موازی و بخش سری هستند. در شکل (۱) یک نمونه از برنامه‌های موازی آمده است. همان‌طور که در این شکل مشاهده می‌شود این برنامه‌ها شامل بخش‌های موازی و سری هستند. به چنین برنامه‌هایی یک وظیفه[۱۲] موازی می‌گوییم.

هر وظیفه موازی می‌تواند به‌صورت یک گراف جهت‌دار بدون دور یا DAG[۱۳] مدل‌سازی شود که در آن هر رأس نشان‌دهنده یک نخ است و هر یال نشان‌دهنده رابطه تقدم[۱۴] بین رأس‌ها است. یک رأس وقتی آماده اجرا است که همه رأس‌های مقدم بر آن اجرا شده باشند. در هر DAG اعداد داخل هر رأس زمان اجرای آن رأس را نشان می‌دهد. مجموع زمان اجرا (بر روی تنها یک هسته) که با پارامتر C مشخص می شود، برابر با مجموع زمان اجرای تمامی راس‌های آن است. در شکل (۲) DAG مربوط به وظیفه شکل (۱) آمده است. در این شکل مجموع زمان اجرا برابر است با $C = 25$.

برای تحلیل زمان‌بندی وظایف نیاز به آگاهی از طول زمان اجرای وظایف بر روی m هسته‌ای است که بر روی آن در حال اجراست. یک روش برای اینکار استفاده از قانون Amdahl[۲۵] است. به این صورت که اگر مجموع زمان اجرای تمامی رأس‌ها (بر روی تنها یک هسته) برابر با C باشد و $f \in [0.1]$ نسبت[۱۵] بخش سریال برنامه مورد نظر باشد، بر اساس قانون Amdahl زمان اجرای موازی این برنامه بر روی m هسته برابر خواهد بود با:

$$C.f + C.\left(\frac{1-f}{m}\right) \qquad (۱)$$

استفاده از این قانون به‌خوبی زمان اجرای وظایف را مشخص می‌کند ولی در این روش نیاز به اطلاع از اندازه بخش سریال برنامه است که معمولاً اندازه‌گیری آن به‌راحتی امکان‌پذیر نیست [۲۶]. یک روش جایگزین در تحلیل‌های زمان اجرا (به جای اندازه‌گیری بخش سری و موازی) استفاده از پارامتر مسیر بحرانی[۱۶] است که در ادامه به آن می‌پردازیم.



شکل ۲- DAG مربوط به برنامه شکل (۱). هر رأس نشان‌دهنده یک نخ و اعداد داخل هر رأس نشان‌دهنده زمان اجرای آن‌ها است. مجموع زمان اجرا برابر با ۲۵ و طول مسیر بحرانی (مشخص شده با خط تیره) ۱۵ است.

۳-۲- مسیر بحرانی

معیار مهم دیگری که در تحلیل زمان اجرای وظایف موازی از آن استفاده می‌شود، مفهوم مسیر بحرانی است. مسیر بحرانی طولانی‌ترین مسیر در DAG وظیفه مورد نظر است و به اندازه آن طول مسیر بحرانی گفته می‌شود و با L نشان داده می‌شود. به عنوان مثال، DAG وظیفه شکل (۲) را در نظر بگیرید. مسیر بحرانی در این DAG برابر است با اندازه طولانی‌ترین مسیر موجود یعنی $L = 1 + 4 + 1 + 7 + 2 = 15$. البته ممکن است طولانی‌ترین مسیر در یک DAG یکتا نباشد. در این شرایط فقط اندازه مسیر بحرانی در تحلیل زمان اجرا مورد نیاز خواهد بود و نه مسیر دقیق آن.

دو پارامتر مجموع زمان اجرا C و طول مسیر بحرانی L در زمان‌بندی وظایف اهمیت فراوانی دارند چراکه پارامتر C برابر است با زمان اجرا وظیفه مورد نظر بر روی تنها یک هسته و پارامتر L نشان دهنده زمان اجرای همین وظیفه بر روی بی‌نهایت هسته است. برای اندازه‌گیری هر دو پارامتر نیز ابزارهای مناسبی وجود دارد. به عنوان مثال هر دو پارامتر مورد نظر در برنامه‌های نوشته شده با Cilk Plus با ابزارهای Cilkview[۲۶] و Cilkprof[۲۷] قابل اندازه‌گیری هستند.

در ادامه برای تحلیل وظایف موازی تنها از دو پارامتر مجموع زمان اجرا و طول مسیر بحرانی استفاده می‌شود. با در نظر گرفتن مفهوم وظایف موازی و مسیر بحرانی، در بخش بعد، مدل سیستم مورد نظر در این پژوهش توضیح داده می‌شود.

۴- مدل سیستم

در این پژوهش سیستم مورد نظر یک سیستم چندپردازنده‌ای با برداشتگر انرژی است. یک مثال کاربردی می‌تواند یک گره «توان صفر[۱۷]» در یک شبکه حسگر بی‌سیم باشد. منظور از گره «توان صفر» گره‌ای است که کاملاً از نظر انرژی خودمختار است و همه انرژی لازم را از محیط، به‌عنوان‌مثال انرژی خورشید، لرزش، اختلاف دما برداشت می‌کند. وظایف این سیستم به‌صورت چندنخی هستند و قابلیت اجرا بر روی چندین هسته را دارند. به هر یک از این وظایف چند نخی که به آن‌ها وظایف موازی می‌گوییم چندین هسته به‌صورت اختصاصی تخصیص می‌یابد. فرض بر این است که سکوی پردازشی دارای n_{total} هسته پردازشی یکسان[۱۸] است که تمامی هسته‌ها دارای سرعت ثابت و برابر هستند. در جدول (۱) توصیف خلاصه‌ای از متغیرها آمده است.

در ادامه مدل سیستم شامل مدل وظایف، انرژی و مخزن ذخیره‌سازی انرژی را معرفی نموده و در انتهای این بخش مسئله مورد نظر را به‌صورت رسمی تعریف می‌کنیم.

جدول (۱) توصیف متغیرها

متغیر	توصیف
τ	مجموعه وظایف سیستم
τ_i	وظیفه i ام
C_i	مجموع زمان اجرای وظیفه τ_i
L_i	طول مسیر بحرانی اجرای وظیفه τ_i
D_i	بیانگر هر دوی موعد و زمان حداقلی بین-ورود وظیفه τ_i
p_i	نرخ مصرف انرژی (یا توان مصرفی) وظیفه τ_i با یک هسته
u_i	بهره‌وری وظیفه τ_i
f_i	نسبت بخش سریال وظیفه τ_i
n_i	تعداد هسته‌های وظیفه τ_i
n_{total}	مجموع تعداد هسته‌های موجود در سکوی پردازشی
P_r	نرخ دریافت انرژی از محیط
B_{max}	حداکثر ظرفیت مخزن ذخیره‌سازی انرژی
B	میزان انرژی موجود در مخزن ذخیره‌سازی در لحظه جاری
n_i^{min}	حداقل تعداد هسته‌های لازم برای رعایت موعد وظیفه τ_i
$\varphi(s)$	حداقل میزان تخصیص انرژی در گام s به وظیفه τ_i
$\omega(s)$	ظرفیت مخزن ذخیره‌سازی اختصاصی وظیفه τ_i
$pd_i(t)$	درخواست پردازشی وظایف پر اولویت‌تر از وظیفه τ_i در بازه زمانی $[0.t]$
$ed_i(t)$	درخواست انرژی وظایف پر اولویت‌تر از وظیفه τ_i در بازه زمانی $[0.t]$
$wd_i(t)$	زمانی که برای تأمین انرژی وظایف پر اولویت‌تر از وظیفه τ_i و خود *وظیفه* τ_i نیاز است.

۴-۱- مدل وظایف

در این سیستم مجموعه وظایف τ شامل n وظیفه موازی پراکنده[۱۹] به‌صورت $\{\tau_1.\tau_2.\dots\tau_n\}$ است که هر وظیفه τ_i شامل چندتایی زیر است:

$$(C_i.L_i.D_i.p_i)$$

- C_i نشان‌دهنده مجموع زمان اجرای وظیفه τ_i است.
- L_i نشان‌دهنده طول مسیر بحرانی اجرای وظیفه τ_i است.
- D_i بیانگر هر دوی موعد[۲۰] و زمان حداقلی بین-ورود است (موعد ضمنی[۲۱]).
- p_i نرخ مصرف انرژی (یا توان مصرفی) وظیفه τ_i با یک هسته است.

در مدل فوق C_i بیانگر زمان اجرای وظیفه τ_i با تنها یک هسته و L_i بیانگر زمان اجرای وظیفه τ_i با بی‌نهایت هسته است. همچنین در زمان ورود هر یک از وظایف اطلاعی از نحوه استفاده آن از هسته‌ها در هر لحظه وجود ندارد و این موضوع در زمان اجرا مشخص می‌شود. به هر نمونه‌ای از وظیفه τ_i نیز یک کار[۲۲] گفته می‌شود. فرض بر این است که وظایف دارای اولویت ثابت هستند و وظیفه با اندیس کوچک‌تر دارای اولویت بالاتری است . یعنی درصورتی‌که دو وظیفه τ_i و τ_j به‌صورتی که $i < j$ داشته باشیم آنگاه τ_i دارای اولویت بالاتری است.

بهره‌وری[۲۳] وظیفه τ_i نیز که با u_i نشان داده می‌شود به‌صورت زیر تعریف می‌شود:

$$u_i = \frac{C_i}{D_i} \qquad (۲)$$

فرض بر این است که این وظایف به دلیل حجم محاسبات بالا دارای بهره‌وری (نسبت زمان اجرا به موعد) بزرگ‌تر از یک هستند یعنی:

$$\frac{C_i}{D_i} \geq 1 \qquad (۳)$$

در این حالت در صورت تخصیص تنها یک هسته به این وظیفه قبل از رسیدن این وظیفه به موعد، نمونه دیگری از این وظیفه ورود[۲۴] کرده است و در نتیجه تخصیص یک هسته برای رعایت موعد کافی نیست.

هر وظیفه τ_i دارای مجموع زمان اجرا و طول مسیر بحرانی مشخص است. با فرض ثابت بودن تعداد هسته‌های تخصیص یافته به این وظیفه، به دلیل عدم قطعیت در نحوه اجرای DAG یک وظیفه موازی و نیز مشخص شدن مسیرهای اجرایی DAG در زمان اجرا (به دلیل وجود شرط‌ها و حلقه‌ها در برنامه)، شاهد زمان‌های اجرای متفاوتی به ازای هر یک از کارهای یک وظیفه خواهیم بود. به‌عبارت‌دیگر با ثابت بودن پارامترهای C_i، L_i و تعداد هسته‌ها، زمان اجرای هر کار از وظیفه τ_i می‌تواند متفاوت باشد. با وجود این دو کران زیر را می‌توان برای زمان اجرا مشخص کرد:

لم ۱) کران بالای زمان اجرای وظیفه τ_i با m هسته برابر خواهد بود با:

$$\left\lceil \frac{C_i - L_i}{m} \right\rceil + L_i \qquad (۴)$$

اثبات:

اگر f_i بخش سری وظیفه τ_i و L_i طول مسیر بحرانی آن باشد رابطه زیر همواره برقرار است:

$$f_i.C_i \leq L_i$$

در نتیجه خواهیم داشت:

$$f_i.C_i \leq L_i \Rightarrow \frac{C_i - f_i.C_i}{m} + f_i.C_i \leq \left\lceil \frac{C_i - L_i}{m} \right\rceil + L_i$$

حال با توجه به اینکه طبق قانون Amdahl فرمول $\frac{C_i - f_i.C_i}{m} + f_i.C_i$ یک کران بالا برای زمان اجرای وظیفه τ_i با m هسته است، $\left\lceil \frac{C_i - L_i}{m} \right\rceil + L_i$ نیز یک کران بالا برای آن خواهد بود. ■

لم ۲) کران پایین زمان اجرای وظیفه τ_i بر روی m هسته برابر است با:

$$\max\left\{\left\lceil \frac{C_i}{m} \right\rceil . L_i\right\} \qquad (۵)$$

اثبات:

کران پایین زمان اجرای وظیفه τ_i همواره L_i است و زمان اجرا حتی با بی‌نهایت هسته نیز از L_i کوچک‌تر نخواهد بود. در شرایطی که وظیفه τ_i شامل m نخ مستقل و موازی باشد زمان اجرا برابر با $\left\lceil \frac{C_i}{m} \right\rceil$ خواهد بود به شرطی که مقدار $\left\lceil \frac{C_i}{m} \right\rceil$ کوچک‌تر از L_i نباشد. لازم به تاکید است که در فرمول (۵) به دلیل وجود تابع max کران پایین زمان اجرا هیچگاه کوچک‌تر از L_i نخواهد بود. ■

در فرمول (۴) درصورتی‌که m بی‌نهایت باشد زمان اجرا برابر با L_i و درصورتی‌که $m = 1$ باشد زمان اجرا برابر با C_i خواهد بود. در ادامه از فرمول (۴) به عنوان کران بالای زمان اجرا استفاده می‌شود.

۴-۲- مدل انرژی

در این قسمت از مدل انرژی مشابه [۲۸] استفاده می‌شود. فرض بر این است که تابع دریافت انرژی از محیط به‌صورت $P_r(t)$ است. این تابع یک تابع ثابت است یعنی $P_r(t) = P_r$. بنابراین انرژی با نرخ ثابت P_r وارد سیستم می‌شود و مجموع انرژی دریافت شده از محیط در بازه زمانی $[t_1. t_2]$ برابر است با:

$$(t_2 - t_1) \times P_r \qquad (۶)$$

در ادامه به جای تابع $P_r(t)$ از P_r استفاده می‌کنیم.

با توجه به اینکه وظایف موازی هستند و می‌توانند با تعداد هسته‌های متفاوتی اجرا شـــوند، بنابراین در درجه اول نرخ مصـــرف انرژی هر وظیفه بســـتگی به تعداد ه سته‌های تخ صیص یافته به آن وظیفه دارد. فرض زیر در مورد م صرف انرژی هر وظیفه وجود دارد:

فرض ۱) فرض بر این ا ست که توان م صرفی وظیفه τ_i با یک ه سته برابر p_i و با m هسته برابر است با $m \times p_i$.

در اینجا بر اســاس فرض ۱ انرژی ایســتای هســته‌ها و ســربار انرژی انتقال کدها و داده‌ها بین هســته‌ها ناچیز در نظر گرفته شــده اســت. به طور دقیق‌تر، فرض شــده است که موارد فوق در پارامتر p_i لحاظ شده است

علاوه بر این، ســیســتم دارای مخزن ذخیره‌ســازی انرژی با حداکثر ظرفیت B_{max} اســت. در هر لحظه، میزان انرژی موجود در مخزن ذخیره‌ســازی با B نشــان داده می شود. فرض بر این ا ست که مشابه [۲۹] زمان‌های ذخیره سازی انرژی و م صرف انرژی ممکن است همپوشانی نیز داشته باشند.

۴-۳-تعریف مسئله

مسئله مورد بررسی در این مقاله به این شرح است:

"ارائه یک زمان‌بندی برای مجموعه وظایف τ با نرخ انرژی ورودی ثابت P_r ، به‌نحوی‌که تمامی وظایف τ موعد خود را رعایت کنند."

در ادامه، برای حل م سئله تعریف شده ابتدا جزئیات بی‌شتری در مورد نحوه اجرای وظایف موازی در هسته‌های اختصاصی معرفی می شود. سپس نحوه مصرف انرژی وظایف موازی برر سی شده و نحوه تأمین انرژی در بدترین حالت معرفی می‌گردد. پس از آن، روشـــی برای تعیین تأخیر تأمین انرژی وظایف موازی ارائه شـــده و با افزایش تعداد هسته‌ها سعی می شود که این تأخیر جبران شود. پس از تمامی این موارد الگوریتم مورد نظر برای زمان‌بندی انرژی وظایف موازی معرفی شـــده و در نهایت نتایج آن در بخش آزمایش‌ها با الگوریتم بهینه مقایسه می‌شود.

۵- زمان‌بندی فدرال

همان‌طور که اشاره شد وظایف دارای بهره‌وری بزرگ‌تر از یک هستند و تخصیص یک هسته برای رسیدن آن‌ها به موعد کافی نیست. در این حالت از یک روش زمان‌بندی فدرال استفاده می‌شود. در این روش زمان‌بندی حداقل تعداد هسته‌های لازم به‌صورت اختصاصی به هر وظیفه تخصیص داده می‌شود. یعنی یک هسته اختصاص‌داده‌شده به یک وظیفه حتی در صورت بیکاری نیز به وظیفه دیگری تخصیص داده نمی‌شود. هر وظیفه داخل مجموعه هسته‌های اختصاصی خود با استفاده از یک زمان‌بند حریصانه زمان‌بندی می‌شود. به این معنی که هر زمان که هر یک از راس‌های DAG وظیفه τ_i آماده اجرا باشد و منابع کافی برای اجرای آن وجود داشته باشد راس مورد نظر اجرا خواهد شد.

در ادامه این بخش ابتدا زمان‌بند حریصانه و سپس نحوه تعیین تعداد هسته‌های هر وظیفه توضیح داده می‌شود.

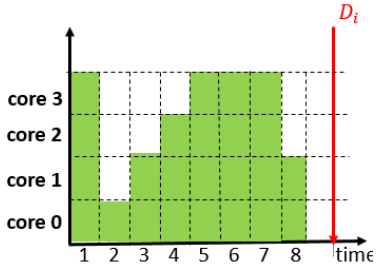
۵-۱- زمان‌بند حریصانه

پرسشی که در این قسمت وجود دارد این است که حال با داشتن پارامترهای C_i، L_i نحوه اجرای وظیفه τ_i در داخل مجموعه هسته‌های اختصاصی خود چگونه خواهد بود و در هر لحظه چه تعداد از هسته‌های اختصاصی آن در حال اجرای وظیفه τ_i خواهند بود و چه تعداد از آن‌ها بیکار خواهند بود؟ برای پاسخ دادن به این پرسش‌ها ابتدا چند تعریف ارایه می‌شود و پس از با استفاده از یک لم به این پرسش پاسخ داده می‌شود.

در این سیستم زمان به بازه‌های گسسته برابری به اسم گام[۲۵] تقسیم شده است. هر واحد زمانی برای هر کار نیز یک گام است. در هر یک از گام‌ها یک هسته می‌تواند بیکار [۲۶] باشد یا یک واحد از کار را انجام دهد. گوییم یک گام به شرطی **کامل** است که هیچ‌کدام از هسته‌های وظیفه τ_i در آن بیکار نباشند، در غیر اینصورت آن گام **غیرکامل** است. یک زمان‌بند حریصانه در صورت وجود کار آماده اجرا یک هسته را بیکار باقی نمی‌گذارد. در شکل (۳) یک مثال از گام‌های کامل و غیرکامل یک وظیفه آمده است.

لم ۳) زمانی که وظیفه τ_i توســط یک زمان‌بند حریصــانه اجرا می‌شــود، هر گام **غیرکامل** یک واحد از طول مسیر بحرانی L_i کم می‌کند[۹].

لم ۳ حالت‌های ممکن برای اجرای وظیفه τ_i را تبیین می‌کند. بر اساس این لم، در طول اجرای وظیفه τ_i حداکثر به تعداد L_i گام غیرکامل وجود خواهد داشت و بقیه گام‌ها الزاما کامل هستند. لذا، علی‌رغم غیرقطعی بودن نحوه اجرای هر وظیفه بر روی هسته‌های اختصاصی خود، همچنان می‌توان حالت‌های ممکن و غیرممکن را برای نحوه اجرا در نظر گرفت. به عنوان مثال، اگر در شکل ۳ اندازه مسیر بحرانی برابر با ۴ باشد، این شکل یک تصور ممکن از نحوه اجرای وظیفه مورد نظر بر روی هسته‌های اختصاصی خود خواهد بود. ولی درصورتی‌که اندازه مسیر بحرانی برابر با ۳ باشد، این نحوه اجرا امکان‌پذیر نخواهد بود چرا که بر اساس لم ۳ تعداد گام‌های غیرکامل نمی‌تواند بزرگتر از طول مسیر بحرانی باشد. در ادامه، در بخش ۶ از مفهوم گام‌های کامل و غیرکامل برای تحلیل بدترین حالت مصرف انرژی نیز استفاده خواهیم کرد.



شکل ۳- مثالی از گام‌های کامل و غیرکامل در زمان اجرای یک وظیفه روی چهار هسته. گام‌های 1,5,6,7 کامل و گام‌های 2,3,4,8 غیر کامل هستند.

۵-۲- تعیین تعداد هسته‌های هر وظیفه

Li و همکاران در [۹] نشان داده‌اند که اگر به وظیفه موازی τ_i با موعد ضمنی D_i تعداد n_i^{min} هسته اختصاصی تخصیص یابد که:

$$n_i^{min} = \left\lceil \frac{C_i - L_i}{D_i - L_i} \right\rceil \qquad (۷)$$

آنگاه در صورت استفاده از یک زمان‌بند حریصانه تمامی کارهای این وظیفه موعد خود را رعایت خواهند کرد. به‌عبارت‌دیگر حداقل تعداد هسته‌های لازم برای رعایت موعد وظیفه τ_i برابر n_i^{min} خواهد بود و با افزایش تعداد هسته‌ها فرصت[۲۷] بیشتری برای اجرای وظیفه τ_i فراهم خواهد بود. همچنین فرض زیر در مورد تعداد هسته‌های یک وظیفه در زمان اجرا وجود دارد:

فرض۲) تعداد هســـته‌ها قبل از شـــروع اجرای هر کار تعیین می‌شـــود و تا انتهای اجرای آن کار نمی‌تواند تغییر کند. به‌عبارت‌دیگر در حین اجرای یک کار نمی‌توان تعداد هسته‌ها را تغییر داد.

بر اساس فرض ۲، تعداد هسته‌های یک وظیفه در اجراهای مختلف می‌تواند متفاوت باشد ولی در حین اجرا تعداد آنها ثابت هستند. در ادامه، نحوه مصرف انرژی وظایف موازی را برر سی می‌کنیم و انرژی هر وظیفه را بر ا ساس بدترین حالت درخوا ست انرژی هر وظیفه تأمین خواهیم کرد.

۶- بدترین حالت مصرف انرژی

یک نکته مهم در مورد وظایف موازی نحوه مصرف انرژی آن‌ها در طول زمان اجرا است. فرض کنیم وظیفه موازی با مجموع زمان اجرای C_i، طول مسیر بحرانی L_i و تعداد m هسته تخصیص یافته است. در این حالت مقدار انرژی درخواستی این وظیفه در هر گام چیست؟

این مسئله در مورد وظایف تک نخی وجود ندارد، چرا که در وظایف تک نخی در هر گام از اجرا فقط از یک هسته استفاده می‌شود و نرخ مصرف انرژی ثابت است. درحالی‌که در وظایف موازی مشخص نیست که در هر گام از اجرا از چه تعداد هسته استفاده می‌شود و در نتیجه میزان انرژی که در هر گام نیاز است قطعی نیست. برای واضح‌تر شدن این مسئله ابتدا با یک مثال این موضوع را بررسی می‌کنیم.

مثال (۱) وظیفه موازی τ_i با پارامترهای زیر را در نظر بگیرید: مجموع زمان اجرا $C_i = 24$، طول مســـیر بحرانی $L_i = 4$، موعد ضـــمنی $D_i = 9$. کارهای این وظیفه در زمان اجرا می‌توانند حالت‌های مختلفی دا شته با شند. حالت‌های مختلف اجرای این وظیفه را با در نظر گرفتن ۴ هسته برای آن برر سی می‌کنیم. در شکل (۴) برخی از حالت‌های اجرا این وظیفه نشان داده شده است.

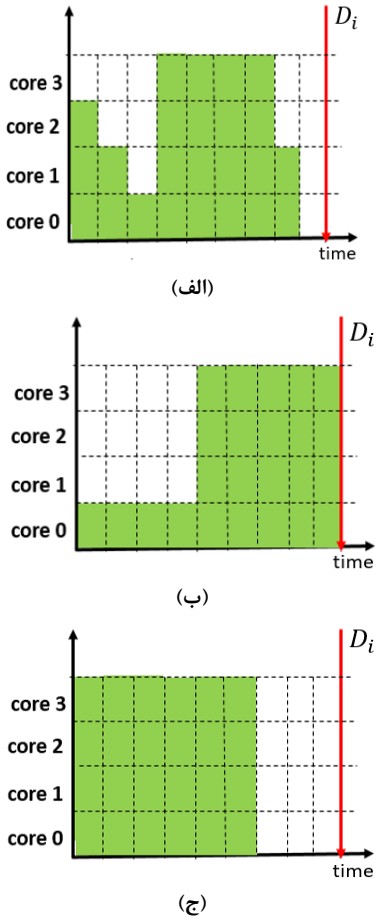
همان‌طور که در شـــکل(۴) ملاحظه می‌شـــود، در حین اجرای وظیفه τ_i بر روی ۴ هســـته به دلیل امکان اســـتفاده از تعداد مختلفی از هســـته‌ها در هر گام، شـــاهد زمان‌های اجرای متفاوتی خواهیم بود و به عنوان نمونه در شـــکل (۴) در گام ۱ ممکن است از یک هسته (حالت ب)، سه هسته (حالت الف) یا چهار هسته (حالت ج) ا ستفاده شود. این متفاوت بودن تعداد هسته‌های ا ستفاده شده در هر گام به دلیل بخش‌های مختلف سری و موازی در کد برنامه ا ست (بر ا ساس لم ۳ حداکثر به اندازه $L_i = 4$ گام **غیرکامل** در حین اجرا می‌تواند وجود داشـــته باشـــد). در ادامه با اســـتفاده از قضـــیه ۱، می‌توان بدترین حالت درخواســـت انرژی یک وظیفه موازی را محاسبه کرد.

قضیه ۱) بدترین حالت درخوا ست انرژی وظیفه موازی τ_i با m هسته به شرطی که $\left\lceil \frac{C_i}{m} \right\rceil \geq L_i$ زمانی اســـت که بیشـــترین موازی‌ســـازی و کمترین زمان اجرا را داشته باشد یعنی زمان اجرای وظیفه برابر باشد با $\left\lceil \frac{C_i}{m} \right\rceil$ باشد.

اثبات:

در هر گام حداکثر میزان درخواست انرژی زمانی خواهد بود که از تمامی هسته‌های تخصیص‌داده شده یعنی m هسته ا ستفاده شود. بدترین حالت درخوا ست انرژی نیز ز مانی خوا هد بود که در تمامی گام ها بیشـــترین میزان انرژی (m وا حد) درخواست شود که در این حالت زمان اجرا برابر خواهد بود با $\left\lceil \frac{C_i}{m} \right\rceil$.

با توجه به کران پایین زمان اجرای مطرح شـــده در فرمول (۵)، مقادیری از m که در آن $\left\lceil \frac{C_i}{m} \right\rceil < L_i$ شود نیز صحیح نیستند و از آن‌ها استفاده نمی‌شود. □



شکل ۴- حالت‌های مختلف استفاده وظیفه موازی مثال(۱) از هسته‌ها. الف- زمان اجرا ۸ واحد ب- زمان اجرا ۹ واحد ج- زمان اجرا ۶ واحد.

بر اساس قضیه ۱ برای اجرای موفقیت‌آمیز وظیفه τ_i که تعداد m هسته به آن تخصیص یافته باید در $\left\lceil \frac{C_i}{m} \right\rceil$ گام اول در هر گام به اندازه m واحد انرژی تأمین شود. با توجه به اینکه در این حالت زمان اجرا برابر با $\left\lceil \frac{C_i}{m} \right\rceil$ خواهد شد و نیز با توجه به بدترین زمان اجرای وظایف که برابر است با $\left\lceil \frac{C_i - L_i}{m} \right\rceil + L_i$، به این نتیجه می‌رسیم که این روش تخصیص انرژی بسیار بدبینانه است و می‌توان آن را با توجه به قضیه ۲ (که در ادامه آمده است) بهبود داد و آن را با توجه به فرصت موجود تا زمان $\left\lceil \frac{C_i - L_i}{m} \right\rceil + L_i$ (کران بالای زمان اجرا) تنظیم کرد.

قضیه ۲) حداقل میزان تخصیص انرژی در گام s به وظیفه τ_i با m هسته، به شرطی که $\left\lceil \frac{C_i}{m} \right\rceil \geq L_i$ باشد و با در نظر گرفتن فرصت تأمین انرژی تا زمان $\left\lceil \frac{C_i - L_i}{m} \right\rceil + L_i$ برابر است با $\varphi(s)$ که تابع φ به‌صورت زیر است:

$$\varphi(s) = \begin{cases} m. & s \leq \left\lceil \frac{C_i - L_i}{m} \right\rceil \\ 1. & \left\lceil \frac{C_i - L_i}{m} \right\rceil < s \leq \left\lceil \frac{C_i - L_i}{m} \right\rceil + L_i \end{cases} \quad (۸)$$

اثبات)

بدترین حالت درخواست انرژی در هر گام برابر با m واحد است که در $\left\lceil \frac{C_i - L_i}{m} \right\rceil$ گام اول به همین میزان تأمین می‌شود. باید نشان دهیم که در گام‌های بعدی اولاً انرژی کافی برای باقیمانده اجرا تأمین شده و ثانیاً زمان کافی برای اجرا با این ترتیب تأمین انرژی وجود دارد.

مجموع انرژی تأمین شده در گام یک تا گام $\left\lceil\frac{C_i-L_i}{m}\right\rceil + L_i$ برابر است با:

$$\sum_{1}^{\left\lceil\frac{C_i-L_i}{m}\right\rceil+L_i} \varphi(s) = m\left\lceil\frac{C_i-L_i}{m}\right\rceil + L_i \geq m\frac{C_i-L_i}{m} + L_i = C_i$$

در نتیجه انرژی کافی در این بازه تأمین می‌شـــود. حال باید دید که آیا زمان کافی برای اجرای تمامی حالات درخواســت انرژی با این روش تأمین انرژی وجود دارد یا خیر؟ باید توجه داشـــت که در هر گام حداکثر به اندازه m واحد از کار را می‌توان اجرا کرد. حال فرض کنیم که در گام $\left\lceil\frac{C_i-L_i}{m}\right\rceil + 1$، درخواست اجرای m واحد از کار وجود دارد. در این گام تنها یک واحد انرژی تأمین می شود و اجرای این مرحله تا m گام بعدی به تأخیر می‌افتد (در هر گام یک واحد انرژی تأمین می‌شود).

حال با توجه به اینکه در هر گام حداکثر به اندازه m واحد از وظیفه مورد نظر را می‌توان اجرا کرد شرایط زیر را خواهیم داشت:

- اگر $m > L$ باشد آنگاه با توجه به این که مجموع کار باقیمانده حداکثر به اندازه L است در نتیجه در یک گام نمی‌تواند به اندازه تمامی m هسته‌ها کار باقیمانده وجود داشته باشد و در نتیجه در گام آخر (یعنی گام $\left\lceil\frac{C_i-L_i}{m}\right\rceil + L_i$) تمامی انرژی مورد نیاز تأمین شده و در همان گام آخر می‌توان آن را اجرا کرد.
- اگر $m \leq L$ باشـد آنگاه در هر گام امکان اسـتفاده از تمامی هسـته‌ها وجود دارد در نتیجه در هر m گام یکبار از تمامی هســـته‌ها اســـتفاده خواهد شد که تعداد این اجراها برابر با L/m خواهد بود.

حال حالتی را در نظر می‌گیریم که در گام‌های نخست حداقل اجرا (تنها یک هسته در هر گام) را داشته باشیم و بیشتر زمان اجرا به گام‌های آخر موکول شده باشند. در این حالت اگر در گام‌های نخست در هر گام فقط از یک هسته استفاده شود که با توجه به لم ۳ طول این گام‌های غیرکامل حداکثر برابر با L خواهد بود و باید در $\left\lceil\frac{C_i-L_i}{m}\right\rceil$ گام بعدی باقیمانده وظیفه را اجرا کرد و دوباره بر ا ساس لم ۳ تمامی این گام‌ها کامل هستند و از تمامی هسته‌ها ا ستفاده می شود. باید نشان داد که از گام $L+1$ تا $\left\lceil\frac{C_i-L_i}{m}\right\rceil + L_i$ در هر گام حداقل m واحد انرژی تأمین شده است و در نتیجه اجراهای هر گام بدون هیچ تأخیر انرژی انجام می‌شود.

باید نشـــان دهیم که در گام s که $L_i < s \leq \left\lceil\frac{C_i-L_i}{m}\right\rceil + L_i$ حداقل m واحد انرژی وجود دارد یعنی:

$$m\left\lceil\frac{C_i-L_i}{m}\right\rceil + \left(s - \left\lceil\frac{C_i-L_i}{m}\right\rceil\right) - L - (s-L-1)m \geq m$$

- $m\left\lceil\frac{C_i-L_i}{m}\right\rceil$: در $\left\lceil\frac{C_i-L_i}{m}\right\rceil$ گام اول در هر گام m واحد انرژی دریافت می‌شود.
- $s - \left\lceil\frac{C_i-L_i}{m}\right\rceil$: مجموع انرژی 1 واحدی دریافت شده.
- L : در L گام اول در هر گام یک واحد انرژی مصرف شده است.
- $s-L-1$: بعد از گام L ام، در هر گام m واحد انرژی مصرف شده است.

حال می‌توان دو حالت را برای s در نظر گرفت:

حالت اول $L < s \leq \left\lceil\frac{C_i-L_i}{m}\right\rceil$

در این حالت با توجه به تابع $\varphi(s)$ در هر گام m واحد انرژی تأمین خواهد شد.

حالت دوم $s > \left\lceil\frac{C_i-L_i}{m}\right\rceil$

با توجه به اینکه در این حالت نرخ ورود انرژی ۱ است و نرخ مصرف انرژی برای با m و $m \geq 1$ بنابراین بدترین حالت کمبود انرژی در گام آخر یعنی گام $s = L + \left\lceil\frac{C_i-L_i}{m}\right\rceil$ رخ می‌دهد:

$$\begin{aligned} m\left\lceil\frac{C_i-L_i}{m}\right\rceil + \left(s - \left\lceil\frac{C_i-L_i}{m}\right\rceil\right) - L - (s-L-1)m \\ = m\left\lceil\frac{C_i-L_i}{m}\right\rceil \\ + \left(\left\lceil\frac{C_i-L_i}{m}\right\rceil + L - \left\lceil\frac{C_i-L_i}{m}\right\rceil\right) - L \\ - \left(\left\lceil\frac{C_i-L_i}{m}\right\rceil + L - L - 1\right)m = m \geq m \end{aligned}$$

□

همان‌طور که ملاحظه شد، قبل از اجرای وظایف اطلاعی در مورد تعداد هسته‌های مورد استفاده در هر گام و در نتیجه انرژی مصرفی آن در هر گام وجود ندارد و تنها می‌توان بدترین حالات را در نظر گرفت (با استفاده از قضیه۲) و بر اساس آن انرژی را به هسته‌ها تخصیص داد. حال این سؤال مطرح می‌شود که درصورتی‌که بدترین حالت درخواســت انرژی اتفاق نیفتد (یعنی در یک گام کمتر از میزان مطرح شـــده در قضیه ۲ انرژی نیاز باشد) باید با انرژی مازاد عرضه شده به یک وظیفه چه کرد؟ پاسخ این است که در این حالت باید انرژی مازاد ذخیره شده و برای گام‌های بعدی همان کار اســـتفاده کرد (در اینجا باید به این نکته توجه داشـــت که مجموع میزان انرژی مصرفی یک کار در طول هر اجرا ثابت است و فقط نحوه مصرف آن می‌تواند متفاوت باشـــد) برای ذخیره‌ســـازی انرژی مازاد یک گام (که به دلیل عدم پیروی از بدترین حالت م صرف انرژی به وجود آمده) از یک مخزن اخت صا صی ذخیره سازی انرژی استفاده می‌شود. اختصاصی بودن مخزن ذخیره‌سازی به این معنی است که انرژی ذخیره شـده در آن فقط مخصـوص همین کار از وظیفه جاری اسـت و از آن نمی‌توان در وظایف دیگر یا حتی کارهای دیگر از همین وظیفه ا ستفاده کرد. علاوه بر این نیازی نیســـت که به ازای هر وظیفه باتری جداگانه‌ای در نظر گرفته شـــود. بلکه می‌توان قســـمتی از ظرفیت باتری را برای هر وظیفه رزرو کرد. این مخزن ذخیره‌ســـازی انرژی فقط در حین اجرا هر کار از وظیفه نیاز خواهد بود و میزان انرژی موجود در آن قبل و بعد از اجرا یک کار صفر ا ست. حال پر سش اینجا ست که حجم این مخزن اختصاصی برای هر وظیفه چقدر ا ست. با ا ستفاده از قضیه ۳، می‌توان به این پرسش پاسخ داد.

قضیه ۳) به ازای هر وظیفه موازی τ_i نیازمند مخزن ذخیره‌سازی انرژی با حداقل ظرفیت زیر هستیم:

$$\omega(s) = \begin{cases} L_i * (m-1). & L_i \leq \left\lceil\frac{C_i-L_i}{m}\right\rceil \\ \left\lceil\frac{C_i-L_i}{m}\right\rceil (m-1). & otherwise \end{cases} \quad (۹)$$

اثبات:

بیشـــترین نیاز به ذخیره‌ســـازی انرژی برای یک وظیفه در زمان اجرا زمانی اتفاق میفتد که بی‌شترین میزان انرژی تأمین شده و کمترین میزان انرژی م صرف شده با شد. با توجه به اینکه تأمین انرژی بر ا ساس تابع $\varphi(s)$ از قضیه ۲ ا ست، در گام $\left\lceil\frac{C_i-L_i}{m}\right\rceil$ مجموع انرژی تأمین شده در ماکزیمم حالت ا ست. حداقل م صرف انرژی نیز زمانی رخ می‌هد که در هر گام فقط یک واحد انرژی مصـــرف شـــود و انرژی باقیمانده در هر گام برابر اســـت با $(m-1)$و تعداد این گام‌ها برابر اســـت با $\min(L_i\ .\left\lceil\frac{C_i-L_i}{m}\right\rceil)$ □

در این بخش نحوه مصرف انرژی وظایف موازی را با داشتن تعداد هسته‌های مختلف بررسی کردیم و نحوه تأمین انرژی هر وظیفه را در بدترین حالت مشخص نمودیم. در بخش بعدی نحوه تعیین تعداد هسته‌ها با هدف جبران تأخیر وظایف پر اولویت‌تر را توضیح می‌دهیم و پس از آن آزمون زمان‌بندی پذیری و الگوریتم زمان‌بندی وظایف، مبتنی بر بدترین روش درخواست انرژی مطرح شده در قضیه ۱، را نیز معرفی می‌کنیم.

۷- زمان‌بندی انرژی

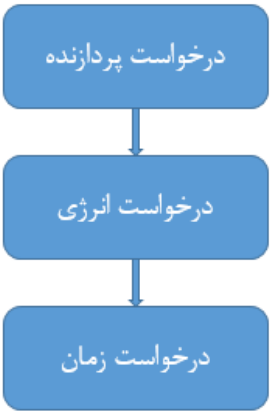
همان‌طور که در بخش قبل توضیح داده شد انرژی با یک نرخ ثابت وارد سیستم می‌شود و ممکن است تأمین انرژی یک وظیفه پر اولویت مانع از تأمین انرژی و اجرای وظایف دیگر شود. در این حالت سؤال اساسی این است که آیا می‌توان با یک استراتژی تأمین انرژی به همراه تعیین مناسب تعداد هسته‌های هر یک از وظایف همگی آن‌ها موعد زمانی خود را رعایت کنند؟ در اینجا منظور از زمان‌بندی انرژی، زمان‌بندی تأمین انرژی هر یک از وظایف است به‌نحوی‌که تمامی وظایف موعد خود را رعایت کنند. برای این کار باید به چند پرسش اساسی پاسخ داد:

- در زمان ورود یک کار از یک وظیفه، تأمین انرژی وظایف پر اولویت‌تر اجرای کار جاری را چه میزان به تأخیر می‌اندازد؟
- آیا با اضافه کردن تعداد هسته‌ها و موازی‌سازی بیشتر می‌توان تأخیر به وجود آمده را جبران کرد تا وظیفه جاری موعدش را رعایت می‌کند؟

در ادامه این بخش ابتدا روشی برای محاسبه تأخیر تأمین انرژی وظایف ارائه می‌شود، پس از آن نحوه تعیین تعداد هسته‌ها با هدف جبران تأخیر ذکر شده ارائه می‌شود و پس از آن آزمون زمان‌بندی پذیری و الگوریتم زمان‌بندی وظایف مبتنی بر بدترین روش درخواست انرژی مطرح شده در قضیه ۱ معرفی می‌شود.

۷-۱- تأخیر تأمین انرژی

برای محاسبه تأخیر تأمین انرژی وظایف لازم است که به ازای هر وظیفه τ_i ابتدا درخواست پردازنده[۲۸] وظایف پر اولویت را تعیین کرد، سپس انرژی لازم برای اجرای این مقدار از درخواست پردازنده را مشخص کرد و پس از آن مقدار زمان لازم برای تأمین این میزان انرژی را تعیین کرد. روند این محاسبات در شکل (۵) امده است. پس از این محاسبات می‌توان تعیین کرد که آیا می‌توان با اضافه کردن تعداد هسته‌های وظیفه τ_i تأخیر به وجود آمده در اجرای آن را جبران کرد؟ در ادامه مفاهیم «درخواست پردازنده»، «درخواست انرژی» و «درخواست زمان» را معرفی نموده و از آنها برای تعیین تعداد هسته‌های هر وظیفه استفاده می‌کنیم.



شکل ۵- روند محاسبه تأخیر انرژی وظایف

تعریف ۱) درخواست پردازشی. درخواست پردازشی وظایف پر اولویت‌تر از وظیفه τ_i برابر است با میزان زمان پردازنده‌ای که وظایف پر اولویت‌تر از وظیفه τ_i و خود *وظیفه* τ_i در بازه زمانی $[0.t]$ درخواست می‌کنند:

$$pd_i(t) = \sum_{j \le i} \left\lceil \frac{t}{D_j} \right\rceil \times C_j \qquad (۱۰)$$

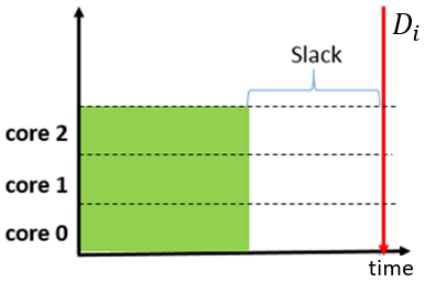
تعریف ۲) درخواست تأمین انرژی . درخواست تأمین انرژی وظایف برابر است با میزان انرژی که برای تأمین انرژی درخواست پردازنده وظایف پر اولویت‌تر از وظیفه τ_i و خود *وظیفه* τ_i در بازه زمانی $[0.t]$ نیاز است که برابر است با:

$$ed_i(t) = \sum_{j \le i} \left\lceil \frac{t}{D_j} \right\rceil \times C_j \times p_j \qquad (۱۱)$$

تعریف ۳) درخواست زمان. درخواست زمان هر وظیفه برابر است با زمانی که برای تأمین انرژی وظایف پر اولویت‌تر از وظیفه τ_i و خود *وظیفه* τ_i نیاز است:

$$wd_i(t) = \left(\sum_{j \le i} \left\lceil \frac{t}{D_j} \right\rceil \times C_j \times p_j \right) / P_r \qquad (۱۲)$$

هدف از این محاسبات تعیین حداقل هسته‌های لازم برای جبران مقدار زمانی است که (به دلیل تأمین انرژی وظایف پر اولویت‌تر) اجرای این وظیفه به تأخیر افتاده است. یعنی با افزایش تعداد هسته‌ها تضمین می شود که اجرای وظیفه τ_i علیرغم تأخیرهای به وجود آمده به دلیل تأمین انرژی سایر وظایف سر موعد انجام خواهد شد.



شکل ۶- فرصت وظایف با افزایش تعداد هسته‌ها افزایش میابد

۷-۲- تعیین تعداد هسته‌ها

همان‌طور که در شکل (۶) ملاحظه می شود، با افزایش تعداد هسته‌های هر وظیفه فرصت آن برای اجرا بیشتر خواهد شد. با توجه به این موضوع، تعداد هسته‌های لازم برای اجرای وظیفه τ_i باید به نحوی باشد که علی‌رغم تأخیر ناشی از تأمین انرژی وظایف پر اولویت‌تر موعد خود را رعایت کند. در این حالت باید میزان تأخیری که در مرحله قبل محاسبه شد را نیز در فرمول تعیین تعداد هسته‌ها گنجاند برای این منظور تعداد هسته‌های هر یک از وظایف برابر خواهد بود با:

$$n_i = \left\lceil \frac{C_i - L_i}{D_i - wd_i(D_i) - L_i} \right\rceil \qquad (۱۳)$$

که این تعداد از هسته‌ها به‌صورت اختصاصی به وظایف تخصیص می‌یابد.

Algorithm 1 Core Assignment

$wd_0 \leftarrow 0$
for $i = 1 \rightarrow n$ **do**
  $j \leftarrow i$
  $ed_i = 0$
  while $j \le i$ **do**
    $ed_i \leftarrow ed_i + \left\lceil \frac{D_i}{D_j} \right\rceil \times C_j \times p_j$
    $j \leftarrow j - 1$
  end while
  $wd_i \leftarrow \frac{ed_i}{P_r}$
end for
for $i = 0 \rightarrow n$ **do**
  if $(D_i - wd_i - L_i) \le 0$
    $n_i \leftarrow \infty$
  else
    $n_i \leftarrow \left\lceil \frac{C_i - L_i}{D_i - wd_i - L_i} \right\rceil$
end for

در Algorithm 1 به ازای هر یک از وظایف تأخیر ناشی از تأمین انرژی وظایف پر اولویت‌تر محاسبه شده و سپس تعداد هسته‌های لازم برای رسیدن به موعد با در نظر گرفتن این تأخیر محاسبه می‌شود. درصورتی‌که میزان این تأخیر به‌اندازه‌ای باشد که امکان رعایت موعد وجود نداشته باشد تعداد هسته‌ها بینهایت در نظر گرفته می‌شود (خط ۱۲ و ۱۳ از Algorithm 1)

۷-۳- زمان‌بندی پذیری

حال که تعداد هسته‌های مورد نیاز وظایف مشخص شده‌اند به بررسی شرایط زمان‌بندی پذیری می‌پردازیم. برای زمان‌بندی پذیری وظایف از الگوریتم زیر استفاده می‌شود.

در این الگوریتم ابتدا بررسی می‌شود که طول مسیر بحرانی کوچک‌تر از موعد باشد. در غیر این صورت با هر تعداد هسته‌ای که به وظیفه مورد نظر تخصیص یابد امکان رسیدن به موعد وجود ندارد. پس از آن تعداد هسته‌های تخصیص یافته از Algorithm 1 مورد بررسی قرار می‌گیرد. همان‌طور که در بخش قبل اشاره شده درصورتی‌که امکان جبران تأخیر ناشی از تأمین انرژی وظایف پر اولویت‌تر با افزایش تعداد هسته‌ها وجود نداشته باشد تعداد هسته‌ها بینهایت در نظر گرفته می‌شود. در خط ۶ الگوریتم نرخ مصرف انرژی یک وظیفه با توجه به هسته‌های تخصیص یافته $n_i \times p_i$ بررسی می‌شود. درصورتی‌که این نرخ مصرف انرژی بزرگتر از مجموع انرژی وارد شده به سیستم به اضافه حداکثر ظرفیت باتری باشد امکان اجرای این وظیفه وجود ندارد. در نهایت مجموع تعداد هسته‌های مورد نیاز با مجموع تعداد هسته‌های موجود مقایسه می‌شود.

Algorithm 2 Schedulability Test

```
1:  For i = 0 → n do
2:    if (D_i ≤ L_i) then
3:      Return False
4:    if (n_i = ∞) then
5:      Return False
6:    if (n_i × p_i = P_r + B_max) then
7:      Return False
8:  end for
9:  if (∑ n_i > n_total) then
10:   return False
11: return True
```

۷-۴- الگوریتم زمان‌بندی

همان‌طور که در بخش قبل توضیح داده شد، پس از آنکه هسته‌هایی هر یک از وظایف به صورت اختصاصی برای آن‌ها تخصیص داده شد، هر یک از وظایف در داخل مجموعه هسته‌های خود به‌صورت فدرال و با استفاده از یک الگوریتم زمان‌بندی حریصانه زمان‌بندی می‌شوند به این صورت که نوبت به زمان‌بندی وظایف می‌رسد. در ادامه برای زمان‌بندی انرژی هر یک وظایف از سیاست مشابه ASAP [۷] استفاده می‌شود.

در این الگوریتم تخصیص انرژی به هر وظیفه بر اساس قضیه ۱ انجام می‌شود (ارائه روش زمان‌بندی با استفاده از روش تخصیص انرژی قضیه ۲ به مقالات آتی موکول می‌شود). در هر گام ابتدا پر اولویت‌ترین وظیفه انتخاب شده و در صورت فعال بودن یک کار از این وظیفه، موجود بودن انرژی برای یک گام اجرای آن بررسی می‌شود و در صورت وجود انرژی کافی (مجموع انرژی ذخیره شده در باتری به اضافه انرژی دریافت شده از محیط) وظیفه مورد نظر به اندازه یک گام اجرا می‌شود. در این الگوریتم در ابتدای در هر گام ابتدا به ازای هر کار فعال درصورتی‌که مجموع انرژی موجود در گام جاری (شامل انرژی دریافت شده از محیط به اضافه انرژی ذخیره شده در باتری) بزرگ‌تر از نرخ مصرف انرژی وظیفه τ_i در این لحظه یا $n_i * p_r$ باشد، آنگاه وظیفه τ_i یک گام اجرا می‌شود. سپس همین روال برای وظایف بعدی نیز تکرار می‌شود. درصورتی‌که در انتهای این گام انرژی مازاد وجود داشته باشد در باتری ذخیره می‌شود. زمانی که انرژی کافی برای اجرای یک وظیفه وجود ندارد، این وظیفه اجرا نمی‌شود و منتظر جمع شدن انرژی در باتری می‌ماند. در بخش بعدی با استفاده از شبیه سازی و انجام آزمایشات، نتایج به‌دست‌آمده از روش ارائه شده با الگوریتم بهینه مقایسه شده است.

Algorithm 3 Para_ASAP Scheduling

```
1:  t ← 0
2:  Loop
3:    E ← P_r + B
4:    for i = 0 → n do
5:      if (τ_i is active )then
6:        if (n_i * p_r ≤ E) then
7:          execute τ_i for one step
8:          E ← E − n_i * p_r
9:        end if
10:     end if
11:   end for
12:   B ← min(E. B_max)
13:   t ← t + 1
14: end loop
```

۸- آزمایش‌ها

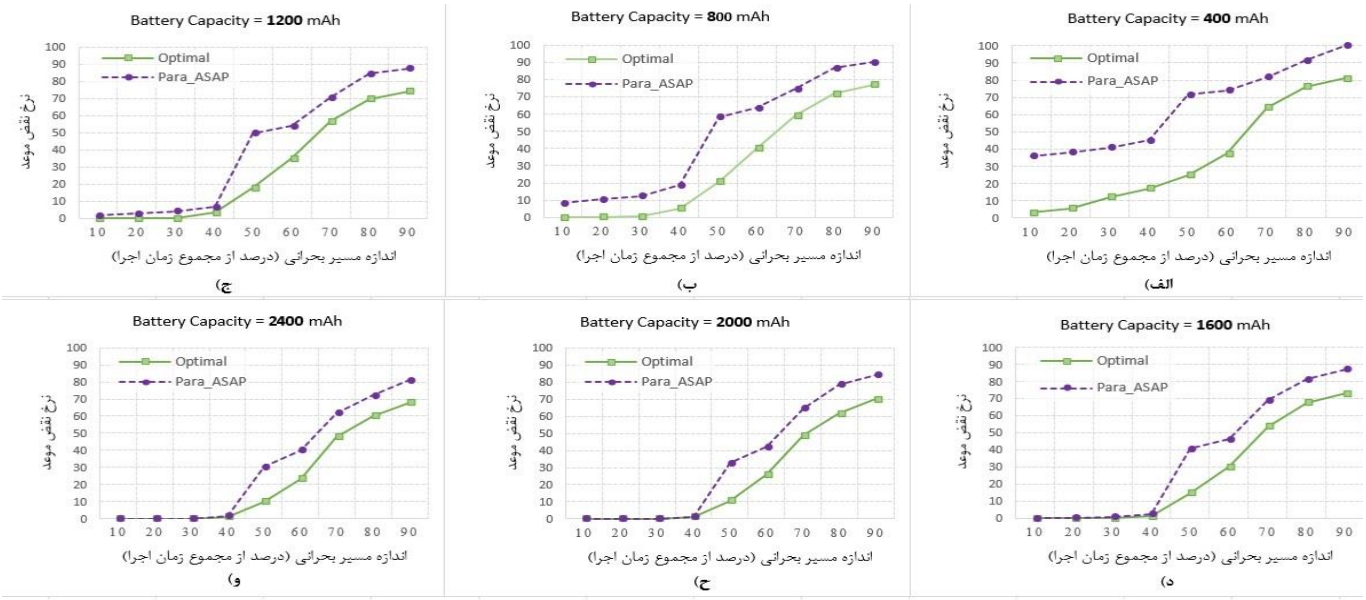
برای بررسی صحت الگوریتم یک شبیه‌ساز برای الگوریتم‌های مورد نظر پیاده‌سازی شد و تأثیر میزان موازی بودن وظایف و نیز نقش باتری در زمان‌بندی پذیری وظایف مورد آزمایش قرار گرفته و نتایج به‌دست‌آمده از روش ارائه شده با حالت بهینه مقایسه شد. در ادامه نحوه تولید وظایف، پارامترهای مرتبط با انرژی، زمان‌بند بهینه و تعداد هسته‌ها معرفی شده و پس از شرح نتایج به تحلیل نتایج به‌دست‌آمده می‌پردازیم.

تولید وظایف: در هر دور از آزمایش مجموعه وظایفی شامل شش وظیفه دوره‌ای تولید شدند که موعد وظایف به‌صورت تصادفی توسط یک توزیع نمایی در بازه [40-2560] تولید شد. مجموع زمان اجرای وظایف به نحوی انتخاب شد که در اجراهای مختلف وظایفی با بهره‌وری معادل ۱۰۰، ۱۲۵، ۱۵۰، ۱۷۵، ۲۰۰، ۲۲۵ و ۲۵۰ درصد تولید شود. اولویت وظایف نیز بر اساس سیاست موعد-یکنواخت[۲۹] انتخاب شد.

پارامترهای انرژی: نرخ مصرف انرژی وظایف به‌صورت تصادفی در بازه [5-60] میلی‌وات تولید شد. پارامتر انرژی ورودی به سیستم P_r به نحوی انتخاب شد که حداقل میزان انرژی لازم برای امکان‌پذیر بودن وظایف تأمین شود. برای این کار مجموع انرژی مصرفی وظایف در طول یک ابر دوره محاسبه شد. سپس P_r برابر با بزرگ‌ترین عدد صحیح حاصل از تقسیم انرژی کل مصرفی یک ابر دوره بر طول ابر دوره قرار داده شد.

زمان‌بند بهینه. با توجه به عدم وجود الگوریتم رقیب، امکان مقایسه نتایج به‌دست‌آمده از الگوریتم Para_asap با کارهای مشابه وجود ندارد. لذا، مشابه [۱۷]، نتایج به‌دست‌آمده از این الگوریتم با خروجی الگوریتم بهینه (که در شکل ۷ با عنوان Optimal مشخص شده است) مقایسه شده‌اند. الگوریتم بهینه با استفاده از جستجوی فراگیر[۳۰] فضای حالت را برای وظایف مختلف به ازای تمامی تعداد هسته‌های ممکن و با ترتیب‌های اجرای متفاوت تا رسیدن تمامی آنها به موعدشان جستجو می‌کند. الگوریتم Optimal دارای پیچیدگی محاسباتی بالایی است (از درجه $O(n^n)$).

تعداد هسته‌ها. مجموع تعداد هسته‌های هر آزمایش نیز در هر دو الگوریتم Para_asap و Optimal یکسان و بر اساس [۳۰] به اندازه $3 \times \lceil \sum_{i=0}^{n} u_i \rceil$ (سه برابر مجموع بهره‌وری تمامی وظایف تولید شده) در نظر گرفته شده است.



شکل ۷- مقایسه الگوریتم ارایه شده با حالت بهینه در ظرفیت‌های مختلف باتری.

بنابراین با توجه به تعداد وظایف و نیز بهره‌وری آنها که در بخش تولید وظایف به آن پرداخته شده، در تمامی آزمایش‌ها تعداد هسته‌ها بین ۶ تا ۱۵ بود.

شرح آزمایش. در این آزمایش وظایفی با طول مسیر بحرانی متفاوت و اندازه باتری‌های مختلف مورد بررسی قرار گرفت. برای این کار مجموعه وظایفی با طول مسیر بحرانی ۱۰ درصد تا ۹۰ درصد تولید و سپس زمان‌بندی پذیری وظایف به ازای ۶ اندازه مختلف باتری مورد بررسی قرار گرفت (باتری با ولتاژ 2.7 ولت مشابه [۳۱] در نظر گرفته شد). در این آزمایش به ازای هر کدام از اندازه‌های مختلف باتری ۲۰۰۰۰ مجموعه وظیفه مختلف با مشخصات ذکر شده مورد بررسی قرار گرفت. در این آزمایش میزان رعایت موعد وظایف با موازی‌سازی‌های مختلف بررسی شد. در هر دور آزمایش وظایف با پیکربندی‌های ذکر شده یک بار با الگوریتم Para_asap و یکبار نیز با زمان‌بند بهینه اجرا شد. نتایج مشاهده شده در شکل (۷) آمده است.

بررسی نتایج. همان‌طور که اشاره شد وظایف با اندازه‌های مختلف مسیر بحرانی، از ۱۰ درصد کل زمان اجرا تا ۹۰ درصد زمان اجرا مورد بررسی قرار گرفتند و شش آزمایش با مقادیر ۴۰۰ تا ۲۴۰۰ میلی‌آمپر ساعت برای ظرفیت باتری انجام شد. همان‌طور که در شکل (۷) مشاهده می‌شود، در الگوریتم Para_asap در تمامی آزمایش‌ها با افزایش نسبت طول مسیر بحرانی به کل زمان اجرا، نرخ عدم رعایت موعد افزایش می‌یابد، به‌صورتی که با افزایش طول مسیر بحرانی از ۴۰٪ زمان اجرا، شاهد کاهش چشمگیر نرخ رعایت موعد در الگوریتم Para_asap نسبت به الگوریتم Optimal هستیم. در قسمت الف شکل (۷) ملاحظه می‌شود که با مقادیر کمتر ظرفیت باتری، الگوریتم Para_asap فاصله بیشتری با الگوریتم Optimal پیدا می‌کند و با افزایش ظرفیت باتری نتایج الگوریتم Para_asap به الگوریتم Optimal نزدیک‌تر می‌شود. همان‌طور که در این شکل‌ها ملاحظه می‌شود، این الگوریتم با ظرفیت مناسب باتری (قسمت **ج** تا **و** شکل (۷)) در وظایفی با مسیر بحرانی کمتر از ۴۰ درصد، افزایش کارایی چشمگیری دارد و نتایج آن نزدیک به نتایج الگوریتم Optimal است. لازم به یادآوری است که اندازه مسیر بحرانی ۱۰۰٪، به معنای سری بودن کامل وظیفه مورد نظر است و با توجه به اینکه بهره‌وری وظایف در آزمایش‌ها بیشتر از ۱۰۰٪ است در این حالت امکان رعایت هیچ موعدی وجود ندارد. به همین منظور وظایف با مسیر بحرانی بیش از ۹۰٪ از نتایج آزمایش‌ها حذف شده‌اند.

۹- نتیجه‌گیری

در این مقاله زمان‌بندی وظایف بی‌درنگ موازی اولویت-ثابت در سیستم‌های سایبر-فیزیکی چندهسته‌ای با محدودیت انرژی مورد بررسی قرار گرفت. در این پژوهش تحلیلی برای بدترین حالت مصرف انرژی وظایف موازی ارائه شد. برای زمان‌بندی انرژی وظایف موازی نیز از این ایده بهره‌برداری شد که تأخیر تأمین انرژی وظایف را می‌توان با افزایش تعداد هسته‌ها جبران کرد. نتایج شبیه‌سازی‌های انجام شده نشان می‌دهد که با کاهش طول مسیر بحرانی یک وظیفه و در نتیجه افزایش بخش موازی وظایف، زمان‌بندی پذیری وظایف به‌صورت چشمگیری کاهش می‌یابد. به‌طوری‌که با کاهش طول مسیر بحرانی به کمتر از ۴۰ درصد، الگوریتم ارائه شده نتایجی نزدیک به حالت بهینه دارد.

مراجع

[1] R. (Raj) Rajkumar, I. Lee, L. Sha, J. Stankovic, “Cyber-physical systems,” *Proc. 47th Des. Autom. Conf. - DAC ’10*, p. 731, 2010.

[2] H. El Ghor, M. Chetto, R. H. Chehade, “A real-time scheduling framework for embedded systems with environmental energy harvesting,” *Comput. Electr. Eng.*, vol. 37, no. 4, pp. 498–510, Jul. 2011.

[3] C. Moser, D. Brunelli, L. Thiele, L. Benini, “Real-time scheduling for energy harvesting sensor nodes,” *Real-Time Syst.*, vol. 37, no. 3, pp. 233–260, Oct. 2007.

[4] C. Moser, D. Brunelli, L. Thiele, L. Benini, “Real-time scheduling with regenerative energy,” *Proc. - Euromicro Conf. Real-Time Syst.*, pp. 261–270, 2006.

[5] M. Chetto, A. Queudet, “A note on EDF schedulingfor real-time energy harvesting systems,” *IEEE Trans. Comput.*, vol. 63, no. 4, pp. 1037–1040, Apr. 2014.

[6] M. Chetto, D. Masson, S. Midonnet, “Fixed priority scheduling strategies for ambient energy-harvesting embedded systems,” *Proc. - IEEE/ACM Int. Conf. Green Comput. Commun. GreenCom*, pp. 50–55, 2011.

[7] Y. Abdeddaïm, Y. Chandarli, D. Masson, “The optimality of PFPasap algorithm for fixed-priority energy-harvesting real-time systems,” *Proc. - Euromicro Conf. Real-Time Syst.*, pp. 47–56, 2013.

[8] R. Y. Tsai, R. K. Lenz, “Real time versatile robotics hand/eye calibration using 3D machine vision,” *Proceedings. IEEE Int. Conf. Robot. Autom.*, pp. 554–561, 1988.

[9] J. Li, J. J. Chen, K. Agrawal, C. Lu, C. Gill, A. Saifullah, “Analysis of Federated and Global Scheduling for Parallel Real-Time Tasks,” *Proc. - Euromicro Conf. Real-Time Syst.*, pp. 85–96, 2014.

[10] D. Moss, P. Pittsburgh, “Scheduling of Frame-based Embedded Systems with Rechargeable Batteries,” *Work. Power Manag. Real-time Embed. Syst. (in conjunction with RTAS)*, pp. 1–8, 2001.

[11] J. Lin, A. M. K. Cheng, “Real-time task assignment in rechargeable

multiprocessor systems," *Proc. - 14th IEEE Int. Conf. Embed. Real-Time Comput. Syst. Appl. RTCSA*, pp. 279–284, 2008.
[12] C. Moser, D. Brunelli, L. Thiele, L. Benini, "Lazy scheduling for energy harvesting sensor nodes," *IFIP Int. Fed. Inf. Process.*, vol. 225, pp. 125–134, 2006.
[13] R. Jayaseelan, T. Mitra, X. Li, "Estimating the worst-case energy consumption of embedded software," *Real-Time Technol. Appl. - Proc.*, pp. 81–90, 2006.
[14] H. El Ghor, M. Chetto, R. H. Chehade, "A real-time scheduling framework for embedded systems with environmental energy harvesting," *Comput. Electr. Eng.*, vol. 37, no. 4, pp. 498–510, Jul. 2011.
[15] Y. Chandarli, Y. Abdeddaïm, D. Masson, "The fixed priority scheduling problem for energy harvesting real-time systems," *Proc. - 18th IEEE Int. Conf. Embed. Real-Time Comput. Syst. Appl. RTCSA - 2nd Work. Cyber-Physical Syst. Networks, Appl. CPSNA*, pp. 415–418, 2012.
[16] K. Faramarzi, M. Hasanloo, M. Kargahi, "The PFPASAP algorithm for energy harvesting real-time systems with a non-ideal supercapacitor," *5th Int. Conf. Comput. Knowl. Eng.*, pp. 279–284, 2015.
[17] M. Shirazi, M. Kargahi, L. Thiele, "Performance maximization of energy-variable self-powered (m, k)-firm real-time systems," *Real-Time Syst.*, vol. 56, no. 1, pp. 64–111, Jan. 2020.
[18] A. Saifullah, S. Fahmida, V. P. Modekurthy, N. Fisher, Z. Guo, "CPU energy-aware parallel real-time scheduling," *Leibniz Int. Proc. Informatics, LIPIcs*, vol. 165, 2020.
[19] H. Nishikawa, K. Shimada, I. Taniguchi, H. Tomiyama, "Energy-aware scheduling of malleable fork-join tasks under a deadline constraint on heterogeneous multicores," *ACM SIGBED Rev.*, vol. 16, no. 3, pp. 57–62, Oct. 2019.
[20] B. Barzegar, H. Motameni, A. Movaghar, "EATSDCD: A green energy-aware scheduling algorithm for parallel task-based application using clustering, duplication and DVFS technique in cloud datacenters," *J. Intell. Fuzzy Syst.*, vol. 36, no. 6, pp. 5135–5152, Jan. 2019.
[21] D. S. Garey, Michael R.; Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.
[22] R. D. Blumofe, C. F. Joerg, B. C. Kuszmaul, C. E. Leiserson, K. H. Randall, Y. Zhou, "Cilk: An efficient multithreaded runtime system," *J. Parallel Distrib. Comput.*, vol. 37, no. 1, pp. 55–69, Aug. 1996.
[23] "Intel® Cilk™ Plus Language Extension Specification." [Online]. Available: https://www.cilkplus.org/sites/default/files/open_specifications/Intel_Cilk_plus_lang_spec_1.2.htm. [Accessed: 05-Sep-2020].
[24] O. Tardieu, H. Wang, H. Lin, "A work-stealing scheduler for X10's task parallelism with suspension," *ACM SIGPLAN Not.*, vol. 47, no. 8, pp. 267–276, 2012.
[25] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," *AFIPS Conf. Proc. - Spring Jt. Comput. Conf. AFIPS*, pp. 483–485, 1967.
[26] Y. He, C. E. Leiserson, W. M. Leiserson, "The Cilkview scalability analyzer," *Annu. ACM Symp. Parallelism Algorithms Archit.*, pp. 145–156, 2010.
[27] T. B. Schardl, B. C. Kuszmaul, I. T. A. Lee, W. M. Leiserson, C. E. Leiserson, "The Cilkprof scalability profiler," *Annu. ACM Symp. Parallelism Algorithms Archit.*, pp. 89–100, 2015.
[28] Y. Abdeddaïm, Y. Chandarli, D. Masson, "The optimality of PFPasap algorithm for fixed-priority energy-harvesting real-time systems," *Proc. - Euromicro Conf. Real-Time Syst.*, pp. 47–56, 2013.
[29] M. Chetto, D. Masson, S. Midonnet, "Fixed priority scheduling strategies for ambient energy-harvesting embedded systems," *Proc. - IEEE/ACM Int. Conf. Green Comput. Commun. GreenCom*, pp. 50–55, 2011.
[30] J. Li, J. J. Chen, K. Agrawal, C. Lu, C. Gill, A. Saifullah, "Analysis of Federated and Global Scheduling for Parallel Real-Time Tasks," *Proc. - Euromicro Conf. Real-Time Syst.*, pp. 85–96, 2014.
[31] A. Mirhoseini, F. Koushanfar, "HypoEnergy: Hybrid supercapacitor-battery power-supply optimization for Energy efficiency," *Proc. -Design, Autom. Test Eur. DATE*, pp. 887–890, 2011.

جمال محمدی مدرک کارشناسی ارشد خود را در رشته علوم کامپیوتر از دانشگاه صنعتی شریف دریافت نموده است. وی هم‌اکنون دانشجوی دکتری نرم‌افزار دانشگاه تهران است. زمینه‌های تحقیقاتی موردعلاقه وی زمان‌بندی آگاه از انرژی وظایف موازی است.
آدرس پست الکترونیکی ایشان عبارت است از:
jmohammadi@ut.ac.ir

مهدی کارگهی در حال حاضر دانشیار گروه مهندسی برق و کامپیوتر دانشگاه تهران است. او مدرک لیسانس خود را در رشته مهندسی کامپیوتر از دانشگاه صنعتی امیرکبیر در سال ۱۹۹۸ و کارشناسی ارشد و دکترای خود را در رشته مهندسی کامپیوتر از دانشگاه صنعتی شریف به ترتیب در سال‌های ۲۰۰۱ و ۲۰۰۶ دریافت کرده است. او همچنین از سال ۲۰۰۳ محقق دانشکده علوم رایانه در پژوهشگاه دانش‌های بنیادی (IPM) بوده است. آدرس پست الکترونیکی ایشان عبارت است از:
kargahi@ut.ac.ir

محمود شیرازی در حال حاضر استادیار دانشکده علوم رایانه و فناوری اطلاعات در دانشگاه تحصیلات تکمیلی علوم پایه زنجان است. او مدرک کارشناسی خود را در رشته علوم کامپیوتر از دانشگاه تبریز، مدرک کارشناسی ارشد خود را در رشته علوم کامپیوتر از دانشگاه تحصیلات تکمیلی علوم پایه زنجان و مدرک دکتری خود را از پژوهشگاه دانش‌های بنیادی (IPM) در رشته علوم کامپیوتر دریافت کرده است. زمینه‌های تحقیقاتی وی در حوزه سیستم‌های بی‌درنگ و پردازش موازی است.
آدرس پست الکترونیکی ایشان عبارت است از:
m.shirazi@iasbs.ac.ir

۱ Preemptive Fixed Priority as Soon as Possible
۲ Energy demand
۳ Energy-aware
۴ Energy-optimal
۵ Energy harvester
۶ Lazy Scheduling Algorithm
۷ Worst-Case Execution Time
۸ Clairvoyant
۹ Earliest Deadline First
۱۰ Preemptive Fixed Priority as Late as Possible
۱۱ Time contention
۱۲ Task
۱۳ Directed Acyclic Graph
۱۴ Precedence
۱۵ Fraction
۱۶ Critical Path
۱۷ Zero power
۱۸ Identical
۱۹ Sporadic
۲۰ Deadline
۲۱ Implicit deadline
۲۲ Job
۲۳ Utilization
۲۴ Release
۲۵ Step
۲۶ Idle
۲۷ Slack
۲۸ Processor demand
۲۹ Deadline-monotonic
۳۰ Exhaustive search

The CSI Journal on Computing Science and Information Technology

The CSI Journal on
Computing Science and
Information Technology
Vol. 18 No.1, 2020

Energy scheduling of fixed-priority parallel real-time tasks in multi-core cyber-physical systems

Jamal Mohammadi[1], Mehdi Kargahi[2], Mahmoud Shirazi[3]

[1,2] School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran

[3] Department of Computer Science and Information Technology, Institute for Advanced Studies in Basic Sciences (IASBS), Zanjan, Iran

Abstract

The increasing computational demand of cyber-physical systems necessitates for the use of multi-core systems. Parallel multithreaded tasks is thus crucial in efficient use of such multicore systems, especially when some sort of time restrictions exist for performing the tasks. On the other hand, many cyber-physical systems work in energy-limited situations and they are supplied through energy harvester where they should run successfully within some limited energy budget. This paper proposes an analysis method to find the cyber-physical system behavior with respect to the task energy requirements under fixed-priority scheduling. It then provides a scheduling algorithm which respects the energy constraints and improves the system schedulability. Simulation results show that parallelizing computations leads to better schedulability, especially when the task critical path lengths are less than 40% of their total computation demand.

Keywords: Fixed-priority Scheduling, Real-Time Systems, Parallel Tasks, Energy Harvesting.