**Precise Scheduling of Mixed-Criticality Tasks on Varying-Speed Multiprocessors**

سید حسین حسینی

**چکيده**

در سیستم‌های مرسوم بی‌درنگ هر پارامتر سیستم با استفاده از یک مقدار مشخص می‌شود. اصولا در این نوع سیستم‌ها برای زمان‌بندی سیستم از بدترین زمان اجرا[[1]](#footnote-1) استفاده می‌شود. در صورت استفاده از بدترین زمان اجرا در سیستم‌های بی‌درنگ، اختلافی میان زمان واقعی اجرا و بدترین زمان اجرا ایجاد می‌شود که در صورتی که از این فاصله به درستی استفاده نگردد منجربه عدم استفاده بهینه از سیستم می‌شود. Mixed-Criticality بیان می‌کند سطوح بحرانی مختلفی برای وظایف موجود در سیستم و همچنین خود سیستم در نظر گرفته شود و با توجه به رفتار سیستم در runtime گذار بین حالت‌های مختلف رخ دهد. این گذار mode switch نام دارد. در این لحظه می‌تواند فرکانس به منظور ارائه خدمت به وظایف افزایش یا کاهش یابد و یا حتی وظایف با سطوح بحرانی پایین از سیستم کنار گذاشته شوند. معمولا دو سطح بحرانی High-Criticality و Low-Criticality برای این نوع سیستم‌ها تعریف می‌شود. هدف این مقاله ارائه زمان‌بندی برای سیستم‌های Mixed-Critical است که چه در حالت Low-Criticality و چه در حالت High-Criticality زمان‌بندی تمام وظایف موجود در سیستم تضمین شود. همچنین این زمان‌بندی از نوع precise می‌باشد، یعنی هیچ یک از وظایف Low-Critical در این سیستم *حذف نخواهند شد*.

کلمات کليدي

سیستم‌های Mixed-Critical، زمان‌بندی precise، fluid scheduling، سیستم‌های چندهسته‌ای

# مقدمه

پارامتر بدترین زمان اجرا که در سیستم‌های بی‌درنگ مرسوم مورد استفاده قرار می‌گیرد، یک مقدار را به صورت بدبینانه به زمان اجرا سیستم نسبت می‌دهد تا پیش‌بینی‌پذیری در این سیستم‌ها برآورده شود. استفاده از این پارامتر باعث عدم استفاده بهینه از هسته‌های پردازشی سیستم می‌گردد. چرا که معمولا در این سیستم‌ها زمان‌ اجرای وظایف کمتر از بدترین زمان اجرا است و فاصله‌ای که میان زمان واقعی و بدترین زمان اجرای وظیفه به وجود می‌‌آید باعث میشود زمان لختی زیادی بر روی هسته‌های پردازشی ایجاد شود. عدم استفاده مناسب از این زمان‌های لختی می‌تواند باعث عدم مدیریت انرژی مصرفی و بیشینه توان مصرفی سیستم[[2]](#footnote-2) شود. برای مدیریت این مورد سیستم‌های Mixed-Critical معرفی شده‌اند. در این پارادایم برای وظایف و همچنین سیستم سطوح criticality مختلفی تعریف می‌شود. در این مقاله برای وظایف دو سطح criticality مختلف که با High-Criticality و Low-Criticality به آن‌ها اشاره خواهد شد استفاده شده‌است. همچنین برای سیستم نیز مشابه با وظایف دو سطح Criticality استفاده می‌شود. هدف از این مقاله، ارائه یک زمان بندی precise برای وظایف Mixed-Critical است به گونه‌ای که تحت هر شرایطی وظایف deadline خودشان را از دست ندهد. در یک سری از مقالات حوزه Mixed-Critical هنگامی که یکی از وظایف High-Critical ددلاین Low-Critical خودش را رد می­نماید به منظور ارائه خدمت کامل به وظایف High-Critical وظایف Low-Critical از سیستم حذف می‌شوند. این مورد می‌تواند باعث کاهش QoS[[3]](#footnote-3) وظایف Low-Critical شود. همچنین ممکن است در سیستم وابستگی بین وظایف وجود داشته باشد و حذف کامل یک وظیفه منجربه اختلال در وظایف دیگر شود. در این مقاله سه زمان­بندی برای این منظور ارائه شده‌است و مقایسه بین این روش­ها به منظور انتخاب روش بهینه انجام شده­است. کارهای انجام شده در این مقاله عبارت است از:

* ارائه سیستم مدل Mixed-Critical برای زمان­بندی این نوع سیستم­ها. در این روش برای ارضا نیازهای سیستم مدل جدیدی ارائه شده است.
* ارائه روش fpEDF-VD با استفاده از الگوریتم fpEDF و مفهوم virtual deadline.
* معرفی دو روش دیگر مبتنی بر fluid scheduling با نام­های MCF-MP و MCF-FR.
* ارزیابی این روش­ها بر اساس نرخ زمان­بندپذیری تحت پارامترهای مختلف تعداد هسته­ها، بهره­وری سیستم و فرکانس نسبی پردازنده.

حال به معرفی سرفصل­های این مقاله می­پردازیم. در فصل 2 به معرفی سیستم مدل اعم از سیستم مدل استفاده شده در این مقاله برای وظایف و هسته­ها خواهیم پرداخت. در فصل 3 الگوریتم fpEDF-VF و تست زمان­بند­پذیری آن را بررسی خواهیم کرد. در فصل 4 دو الگوریتم fluid scheduling را شرح خواهیم داد و درنهایت در بخش 5 به ارزیابی الگوریتم­های ارائه شده خواهیم پرداخت.

# مدل سیستم و شرح مسئله

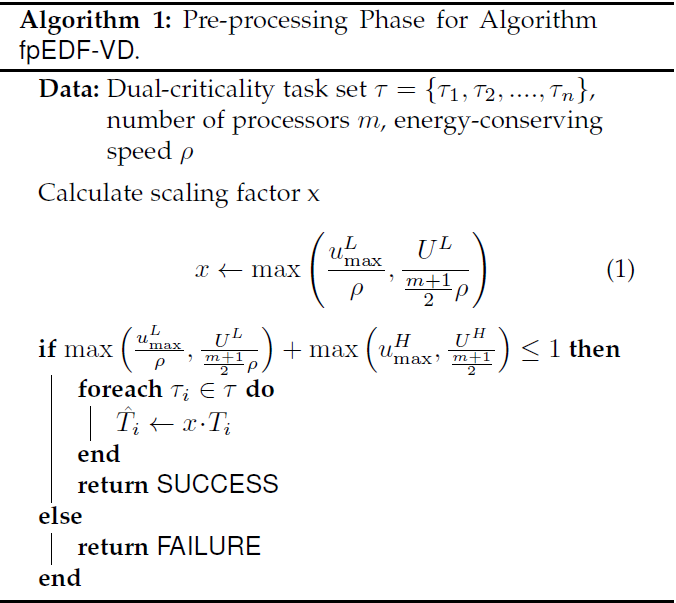
در این سیستم n وظیفه sporadic در نظر گرفته می­شود که با مجموعه نمایش داده خواهند شد. همچنین هر وظیفه با سه­تایی نمایش داده خواهد شد. در این سه تایی برابر با پریود وظیفه، برابر با زمان اجرای low-critical وظیفه و برابر با زمان اجرای high-critical وظیفه می­باشد. در وظایف Low-Critical، می­باشد. همچنین در این سیستم حال کار به دو نوع H-mode و L-mode تقسیم می­شود. H-mode زمانی است که یکی از وظیف High-Critical بودجه زمانی خود را رد نمایند. در حالت کاری L-mode تمامی پردازنده­ها با نسبت فرکانس تقلیل­یافته کار می­کنند و در صورتی که حالت کار سیستم تغییر نماید به منظور خدمت رسانی به وظایف تمامی پردازنده­ها با حداکثر فرکانس خود کار خواهند کرد. مشخص است وظایف Low-Critical قابلیت تغییر حالت کار سیستم را ندارند. همچنین همانطور که ذکر شد هیچ یک از وظایف تحت هیچ شرایطی حذف نخواهند شد. در این سیستم نرخ بهره وری[[4]](#footnote-4) وظایف در حالت H-mode و همچنین L-mode به ترتیب با و نمایش داده می­شود. همچنین بهره­وری تمامی وظایف سیستم در حالت­های H-mode و L-mode به تریب با و نمایش داده خواهند شد.

**شرح مسئله.** هدف در این سیستم این است که با در اختیار داشتن m هسته پردازشی و n وظیفه مطابق آنچه در بالا ذکر شده­است یک مکانیزم زمان­بندی ارائه شود که شرایط زیر را برقرار نماید:

* در طول فاز عملیاتی سیستم، **هیچ** یک از وظایف تحت هیچ شرایطی حذف نشوند. به عبارت دیگر QoS کامل به تمامی وظایف موجود در سیستم ارائه شود.
* هنگامی که سیستم در حالت کاری L-mode می­باشد، تمامی پردازنده­ها به سرعت تقلیل یافته کار می­کنند. در صورتی که یکی از High-Critical بودجه زمانی را رد نماید، حالت کار سیستم به H-mode تغییر می­یابد و تمامی هسته­ها به اجرای وظایف با حداکثر سرعت می­پردازند. بازگشت به L-mode زمانی اتفاق می­افتد که هیچ وظیفه­ای در سیستم باقی نمانده باشد.

# زمان­بندی با virtual deadline

هدف از زمان­بندی با virtual deadline این است که زمانی که وظایف High-Critical سیستم در حالت L-mode در حال اجرا هستند، یک virtual deadline به آن­ها اختصاص داده شود. به این منظور که در صورتی که یکی از وظایف بودجه زمانی خودش را رد کرد، زمان اجرای کافی برای فاز H-mode داشته باشد. در واقع در صورت تغییر حالت سیستم از L-mode به H-mode، مقدار زمان باقی­مانده برای هر وظیفه تا ددلاینش مشخص نمی­باشد و با استفاده از virtual deadline می­توان حداقل زمان را برای این مورد تعیین نمود. همچنین virtual deadline برای وظایف Low-Critical برابر با ددلاین اصلی آن­ها در نظر گرفته می­شود. از این طریق با عقب بردن ددلاین وظایف High-Critical به نوعی اولویت اجرا برای آن­ها بالاتر برده می­شود و زودتر از وظایف Low-Critical انجام می­گردند. در این سیستم ددلاین همه وظایف High-Critical به یک نسبت که با x نمایش داده می­شود عقب می­شود. یعنی مقدار relative virtual deadline برای تمام وظایف High-Critical برابر با می­باشد. مقدار x مطابق با الگوریتم 1 محاسبه می­شود. در صورتی که این الگوریتم مقدار SUCCESS را بگرداند به این معنا است که با توجه به سرعت پردازنده در حالت L-mode مجموعه وظایف سیستم قابل زمان­بندی هستند. زمان­بندی وظایف سیستم در حالت L-mode و H-mode با استفاده از الگوریتم fpEDF انجام می­گیرد [1]. با توجه به اینکه این الگوریتم از مفهوم virtual deadline در کنار الگوریتم fpEDF استفاده می­کند، fpEDF-VD نامیده شده است. نکته منفی که در الگوریتم fpEDF-VD مشخص می­باشد این است که هنگام ورود به L-mode ددلاین وظایف را برابر با بیشینه مقدار ممکن در نظر می­گیریم. همچنین فرض می­شود وظیفه هیچ زمان اجرای در L-mode نداشته است. همین فروض بدبینانه باعث کاهش کارایی سیستم می­گردد. از مزایای این سیستم این است که قابلیت پیاده­سازی بر روی پلتفرم­های مختلف را دارا می­باشد. همچنین هزینه context switch نسبت به رو­ش­های دیگر کمتر است. در ادامه دو الگوریتم با استفاده از fluid scheduling معرفی شده­اند تا این فرض­های بدبینانه را کاهش دهند.



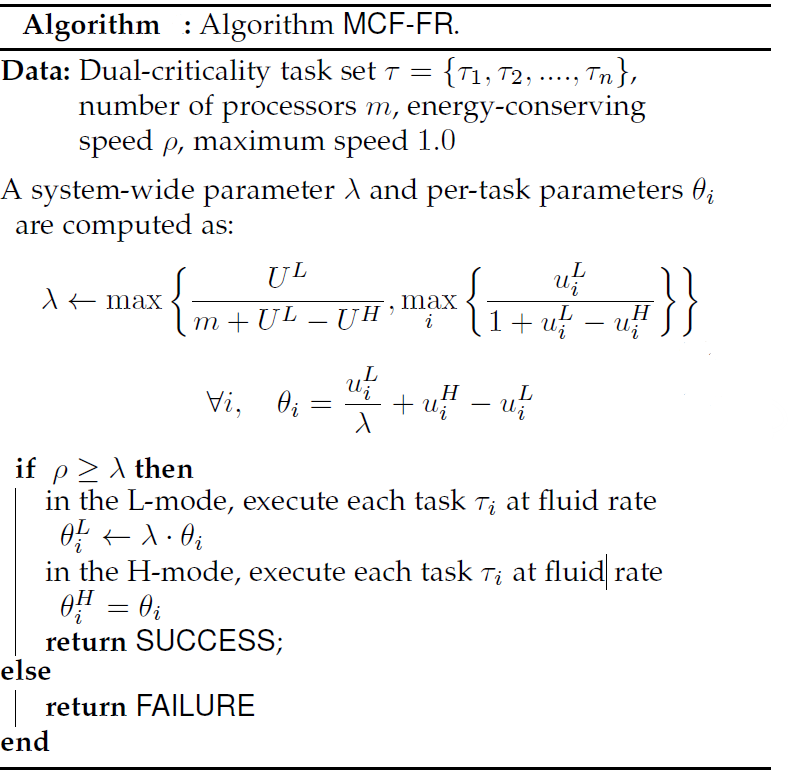
الگوریتم 1. محاسبه virtual deadline

# Dual-Rate Fluid Scheduling

در این قسمت مفهوم Fluid Scheduling را معرفی می­نماییم. در این دسته از الگوریتم­های فرض می­شود هسته­های پردازشی به صورت منصفانه ای به وظایف اختصاص داده می­شود. یکی از روش­های پیاده­سازی این الگوریتم استفاده از اجرای همروند وظایف می­باشد. حال در این قسمت میخواهیم با استفاده از این مفهوم زمان­بندی را انجام بدهیم. با استفاده از این مفهوم به هر وظیفه 2 مقدار تخصیص داده می­شود. و . این مقادیر به این معنا هستند که یک وظیفه در طول وجودش در سیستم در حالت L-mode، به صورت مداوم نسبت از توان پردازشی سیستم را در اختیار می­گیرد و به اجرای وظیفه­اش می­پردازد. نیز برای همین مفهوم در حالت H-mode می­باشد. حال با استفاده از این مقادیر مسئله را به صورت یک مسئله optimization مدل می­کنیم. محدودیت­های ما برای این مقدار به شرح زیر است.

* نسبت­های داده شده نمی­توانند از توان پردازشی سیستم که برابر با m می­باشد عبور کنند. همچنین با توجه به آنکه احتمال شکستن یک وظیفه و اجرای موازی آن وجود ندارد نسبت تخصیص یافته به هر وظیفه باید کمتر از 1 باشد.
* این نسبت­ها می­باید امکان اجرای وظایف را ایجاد کنند. یعنی چه در حالت L-mode و چه در حالت H-mode اجرای وظایف با استفاده از این نسبت­ها تضمین گردد.

در نهایت با مدل کردن مسئله مطابق شروط بالا، به یک مسئله optimization می­رسیم. درصورتی که این مسئله جواب داشته باشد می­توان با استفاده از fluid scheduling، مدل سیستمی که در قسمت 2 به آن اشاره کردیم را زمان­بندی کرد. این الگوریتم را MCF-MP می­نامیم. همانطور که مشخص است مقادیر و برای هر وظیفه به صورت مجزا تعریف می­شوند. این موضوع باعث می­شود تا به یک مسئله optimization غیرخطی برسیم که زمان محاسبه نتیجه آن ممکن است طولانی شود. به همین منظور الگوریتم MCF-FR را معرفی می­نماییم که نسبت و را برای تمامی وظایف یکسان در نظر می­گیرد با عبارت دیگر برای هر وظیفه  
 می­باشد. این فرض باعث خطی شدن مسئله optimization می­گردد. با استفاده از الگوریتم 2 می­توان این مسئله را حل نمود. با استفاده از دو الگوریتم امکان زمان­بندی مجموعه وظایف افزایش می­یابد. منتها با توجه به اینکه برای این نوع اجرای وظایف نیاز به اجرای همروند داریم، هزینه context switch های زیادی در سیستم به ما تحمیل می­گردد. در قسمت بعدی به مقایسه این سه روش از نظر نرخ زمان­بند­پذیری می­پردازیم.

**

الگوریتم 2. MCF-FR

# ارزیابی روش­های ارائه شده

در این قسمت برای حالات مختلف سیستم اعم از تعداد هسته­ها، فرکانس هسته­ها در حالت L-more و نرخ بهره­وری وظایف به مقایسه 3 روش ارائه شده می­پردازیم. برای ایجاد وظایف در سیستم از الگوریتم معروف UUniFast استفاده شده­است. این الگوریتم برای n وظیفه و بهره­وری کلی u برروی یک هسته مورد استفاده قرار می­گیرد. برای m هسته، نرخ بهره­وری کلی سیستم می­تواند حداکثر m باشد. برای این منظور با استفاده از الگوریتم UUniFast، n وظیفه با نرخ بهره­وری به ازای هر هسته[[5]](#footnote-5) را مشخص می­کنیم و تمامی نرخ­های به دست آمده را در m ضرب می­کنیم. در صورتی که تمامی این نرخ­ها از 1 کمتر بودند وظایف ما معتبر هستند. در غیر این صورت وظایف معتبر نمی­باشند. حال پس از تولید وظایف سیستم، وظایف را بر روی سه الگوریتم تست می­نماییم. در نمودار 1 نرخ زمان­بند­پذیری این سه الگوریتم به ازای نرخ بهره­وری­های مختلف به ازای هشت هسته آورده شده است. همان­طور که مشخص است دو الگوریتم MCF-FR و MCF-MP عملکرد به مراتب بهتری را نسبت به fpEDF-VD دارند. دلیل این امر این است که الگوریتم fpEDF-VF یک باند بالا برای بهره­وری تعریف می­کند که مقدار آن برابر با می­باشد. درحالی که دو الگوریتم دیگر برای مجموعه وظایف با نرخ بهره­وری بالا نیز قابل استفاده می­باشند. همچنین الگوریتم MCF-MP در نرخ بهره­وری 0.9 عملکرد بهتری نسبت به MCF-FR داشته است. این نشان می­دهد برای نرخ­های بهره­وری خیلی بالا MCF-MP مناسب­تر می­باشد.

علاوه بر ارزیابی فوق، یک بار نیز سیستم را برای نرخ متوسط ارزیابی می­کنیم. در این قسمت انتظار می­رود با توجه به اینکه سرعت اجرای وظایف در حالت L-mode کاهش می­یابد، نرخ زمان­بندپذیری سیستم کم شود. در نمودار 2 برای سیستمی با 4 هسته و = 0.6 نرخ زمان­بند­پذیری نمایش داده شده­است. همان­طور که مشخص است در الگوریتم­های MCF-FR و MCF\_MP این نرخ وابستگی شدیدی به دارد و با کاهش این مقدار نرخ زمان­بند­پذیری وظایف کاهش محسوسی را نسبت به نمودار 1 داشته است، اما در نتیجه الگوریتم fpEDF-VD تغییر محسوسی مشاهده نمی­شود.

نمودار 1. نرخ زمان­بندپذیری به ازای m =8 و = 0.9

نمودار 2. نرخ زمان­بندپذیری به ازای m = 4 و = 0.6

# نتیجه­گیری

در این مقاله سه روش برای زمان­بندی وظایف Mixed-Critical ارائه شده است. روش اول با نام fpEDF-VD با ترکیب الگوریتم fpEDF و همچنین مفهوم virtual deadline به یک زمان­بند برای این نوع سیستم­ها دست­یافته است. ضعف اصلی این روش این است که به منظور آنکه زمان­بندپذیری سیستم را تضمین نماید، کران بالای بسیار کمی را برای بهره­وری وظایف قرار می­دهد. بنابراین برای وظایف با بهره­وری بالا این روش مناسب نمی­باشد. دو روش دیگر با نام­های MCF-MP و MCF-FR از fluid scheduling به منظور زمان­بندی استفاده می­نمایند که برای مجموعه وظایف با بهره­وری بالا نیز مناسب می­باشند. چالش اصلی استفاده از این دو روش این است که برای اجرای منصفانه وظایف نیاز به اجرای همروند در سیستم می­باشد که سربار context switch زیادی را به سیستم وارد می­کند. کارهایی که در آینده در امتداد این مقاله می­توان انجام داد مدیریت حداکثر توان در این سیستم است. با توجه به آنکه در H-mode تمامی وظایف با حداکثر سرعت خودشان کار می­کنند شاهد مصرف حداکثر توان زیادی در سیستم خواهیم بود.

# مراجع

[1] S. K. Baruah. Optimal utilization bounds for the fixed-priority scheduling of periodic task systems on identical multiprocessors. IEEE Transactions on Computers, 53(6):781–784, 2004.

1. Worst-Case Execution Time (WCET) [↑](#footnote-ref-1)
2. Peak Power [↑](#footnote-ref-2)
3. Quality of Service [↑](#footnote-ref-3)
4. utilization [↑](#footnote-ref-4)
5. Per core utilization [↑](#footnote-ref-5)