علوم رایانش و فناوری اطلاعات
نشریه علمی انجمن کامپیوتر ایران
مجلد ۱۴، شماره ۲، ۱۳۹۵
صفحات ۳۰- ۳۸
**مقاله عادی**

# مدیریت سیستمی دمای پردازنده‌های چندهسته‌ای برای زبان‌های موازی مبتنی بر زمانبند ربایش کار

حمید گوهرجو[۱] مرتضی مرادی[۱و۲] حمید نوری[۱و۲]

[۱]دانشکده مهندسی، دانشگاه فردوسی مشهد، مشهد، ایران
[۲]پژوهشکده علوم کامپیوتر، پژوهشگاه دانش‌های بنیادی (IPM)، تهران، ایران

## چکیده

در سال‌های اخیر، دمای بالا و توان مصرفی زیاد در پردازنده‌های چندهسته‌ای به یک چالش اساسی برای سازندگان و کاربران این پردازنده‌ها تبدیل شده است. با رشد دمای پردازنده، هزینه‌های خنک‌سازی و مصرف توان افزایش یافته و طول عمر پردازنده کاهش می‌یابد. در این تحقیق، یک الگوریتم مدیریت دمای پویا در سطح سیستم عاملپیشنهاد شده است که در اجرای برنامه‌های موازی مبتنی بر زمانبند ربایش کار[۱]، دمای پردازنده را در محدودیت درخواستی کاربر مدیریت می‌کند. از این رو، ما دو مدل دمایی و کارایی را جهت پیش‌بینی دمای آینده و تخمین میزان تغییرات کارایی برنامه پیشنهاد دادیم. با استفاده از مدل‌های پیشنهادی، الگوریتم پیشنهادی تعداد هسته‌های فعال و فرکانس پردازنده را به نحوی تعیین می‌کند که دما از محدودیت تعیین شده پایین‌تر نگه داشته شده و کمترین آسیب ممکن به کارایی برنامه وارد گردد. آزمایشات بر روی سیستم واقعی نشان داد که الگوریتم پیشنهادی به طور میانگین ۲۸ درصد کارایی بالاتری از الگوریتم آگاه از همسایگی داشته و برخلاف این الگوریتم، هرگز از محدودیت دمایی تعیین شده تخطی نمی‌کند.

**کلمات کلیدی:** اجرای موازی، پردازنده چندهسته‌ای، ربایش کار، زمانبند، سیستم عامل، مدیریت دما.

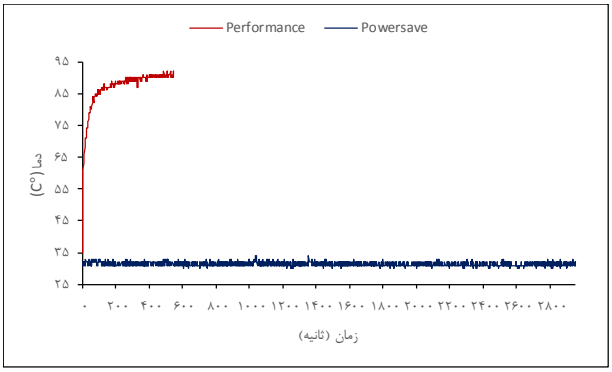
## ۱- مقدمه

در سال‌های اخیر با افزایش تعداد و چگالی ترانزیستورها بر روی تراشه، توان مصرفی و دمای پردازنده‌های چندهسته‌ای افزایش یافته است. با رشد دمای پردازنده، هزینه‌های خنک‌سازی و مصرف توان افزایش یافته و طول عمر پردازنده کاهش می‌یابد [۱]. در مواجهه با این مشکل، راهکارهای مختلفی با محوریت مدیریت دمای پردازنده مطرح شده‌اند که از این میان، دسته‌ای با عنوان مدیریت پویای پیشگیرانه‌ی دما[۲]، رسیدن پردازنده به آستانه‌ی دمایی نامطلوب را با استفاده از مدل‌هایی پیش‌بینی کرده و با انجام اقدامات پیشگیرانه‌ای مانند کاهش فرکانس ساعت پردازنده[۳]، مهاجرت پردازه‌ها[۴] و کاستن از هسته‌های فعال پردازنده ضمن ممانعت از کاهش شدید کارایی از بروز دمای نامطلوب اجتناب می‌کنند [۳].

با ظهور پردازنده‌های چندهسته‌ای، برنامه‌ها ناگزیرند تا با استفاده از زبان‌های برنامه‌نویسی موازی پیاده‌سازی شوند تا بتوانند کارایی بالایی را در اجرا بر روی پردازنده‌های چندهسته‌ای بدست آورند. تعداد زیادی از زبان‌های موازی مانند OpenMP، Intel Cilk Plus و Intel TBB، از زمانبند ربایش‌کار برای ایجاد موازی‌سازی در سطح برنامه بهره می‌برند [۲]. این در حالی است که سیستم عامل‌ها از وجود چنین زمانبندی درون برنامه اطلاعی ندارند. در اجرا به وسیله زمانبند ربایش کار، چندین نخ به‌عنوان کارگر[۵] در سطح سیستم عامل ایجاد می‌شود که تمامی آن‌ها به صورت همزمان به اجرای وظایف تعریف شده در برنامه می‌پردازند. از آنجایی که تخصیص وظایف به نخ‌های کارگر در زمان اجرا به صورت پویا و بدون پیش‌فرضی در برنامه صورت می‌گیرد، در صورت توقف یکی از نخ‌های کارگر، دیگر نخ‌های فعال در برنامه به اجرای وظایف باقیمانده خواهند پرداخت. این در حالی است که در برخی مدل‌های برنامه‌نویسی دیگر در صورت توقف یک نخ ممکن است ادامه‌ی اجرای کل برنامه به چالش کشیده شود. در [۴] و [۵] به بهبود صف نگهداری وظایف در ربایش کار پرداخته شده است. و [۶]، [۷] و [۸]

مسئله‌ی نحوه‌ی انتخاب صف قربانی را جهت رعایت وظیفه بهبود داده‌اند. در [۹] در هنگام رعایت وظیفه به جای یک وظیفه، نصف وظایف موجود در صف قربانی رعایت می‌شود. در [۱۰] رعایت‌کننده خودش حق برداشتن وظیفه از صف قربانی را ندارد بلکه درخواست خود را به قربانی ارسال می‌کند و قربانی در صورت داشتن وظیفه آن را برای رعایت‌کننده ارسال می‌کند. در [۱۱] با استفاده از ساختمان داده لیست پیوندی، یک صف با اندازه پویا ارائه شده است. در [۱۲] به زمان‌بندی وظایف با درنظر گرفتن اولویت در رعایت کار پرداخته شده است. در هیچ کدام از این کارها مساله دما مورد توجه قرار نگرفته است.

در این تحقیق، ما اجرای موازی برنامه‌ی ضرب استاندارد ماتریس را بر روی پردازنده Intel Core i7-4790K توسط دو مدیریت فرکانس powersave و performance موجود در سیستم عامل لینوکس آزمایش کردیم. نمودار ارائه شده در شکل ۱، دمای پردازنده را در این آزمایش نشان می‌دهد که در آن محور افقی زمان اجرا و محور عمودی دمای پردازنده است. همان‌طور که در شکل دیده می‌شود، دمای پردازنده در مدیریت با powersave به طور میانگین حدود ۵۵ درجه کمتر از میانگین دمای performance است. در مقابل زمان اجرای performance تقریباً یک‌چهارم زمان اجرای powersave است. بر این اساس، دریافتیم که با امکانات موجود در سیستم عامل لینوکس نمی‌توان با صرف هزینه کارایی پایینی، دمای پردازنده را در محدوده‌ی دلخواه مدیریت کرد.




شکل ۱- دمای پردازنده تحت مدیریت فرکانس در سیستم عامل لینوکس

تا کنون تحقیقات زیادی برای مسئله‌ی مدیریت پویای دمای پردازنده‌ی چندهسته‌ای صورت گرفته است. در این کارها تمرکز روی معیارهایی از جمله کاهش میانگین دمای پردازنده، کاهش رخداد نقاط داغ[۶] بر روی پردازنده و نیز کنترل دما زیر محدودیت درخواستی کاربر بوده است [۱۳-۱۸]. از میان کارهای صورت گرفته، [۱۳] و [۱۴] عملکرد بهتری از دیگران داشته‌اند. در این کارها، از مدل‌های پیش‌بینی دمای پردازنده بهره‌برده شده است تا دمای آینده‌ی پردازنده پیش‌بینی شود و در صورت لزوم برای اجتناب از گذر دما از محدودیت درخواستی کاربر، عمل پیش‌دستانه‌ای صورت می‌گیرد. تحقیقات دیگری مشابه این کارها صورت گرفته است اما در اکثر این کارها ویژگی‌هایی از جمله موازی‌سازی در برنامه‌های کاربردی مورد توجه قرار گرفته نشده است و لذا کارایی آن‌ها برای چنین کاربردهایی مورد تردید است.

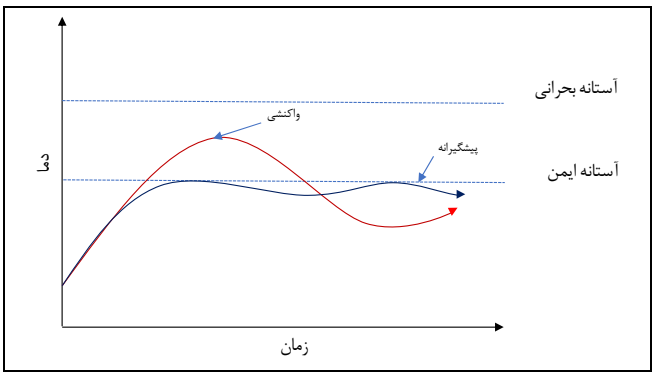
اطلاعات ما نشان می‌دهد که تاکنون هیچ یک از راهکارهای مدیریت دمای مطرح شده در سطح سیستم عامل، آگاهی از وجود زمانبند رعایت کار را در برنامه‌های موازی در نظر نگرفته‌اند. در حالی که با وجود این زمانبند، امکان تغییر تعداد نخ‌های فعال در زمان اجرا به صورت پویا و بدون نیاز به تغییر ساختار برنامه وجود دارد. از این‌رو، در این تحقیق یک الگوریتم مدیریت دمای پویا در سطح سیستم عامل را پیشنهاد دادیم که در اجرای برنامه‌های موازی، دمای پردازنده را کمتر از آستانه‌ی درخواستی کاربر مدیریت می‌کند. در الگوریتم ما، فرض بر این است که برنامه‌ی موازی در حال اجرا با استفاده از زمانبند رعایت کار پیاده‌سازی شده است و در اثر کاهش و یا افزایش تعداد نخ‌ها، روند اجرای برنامه مختل نخواهد شد. در ادامه، نوآوری‌های این پژوهش به صورت خلاصه بیان شده است:

- ارائه الگوریتم مدیریت دما در سطح سیستم عامل برای برنامه‌های موازی مبتنی بر زمانبند رعایت کار برای اولین بار به نحوی که تضمین‌کننده‌ی عدم‌تجاوز دما از سطح درخواستی کاربر باشد.
- پیشنهاد دو مدل دمایی و کارایی با دقت مناسب جهت استفاده در الگوریتم‌های مدیریت دما.
- بررسی دقت مدل دمایی و تاثیر افزایش دقت مدل‌ها در الگوریتم مدیریت دما، بر کارایی برنامه و دمای پردازنده.
- عدم‌تجاوز از محدودیت دمایی کاربر با صرف هزینه‌ی کارایی پایین‌تر از دیگران، چنانچه در کارهای دیگران با وجود تخطی از محدودیت دمایی تعیین شده، کارایی پایین‌تری بدست آمده است.

در ادامه، بخش ۲ پیشینه‌ی تحقیق و کارهای انجام شده در زمینه‌ی مدیریت دما را بررسی کرده است. سپس، مدل‌های دمایی، مدل کارایی و الگوریتم مدیریت دمای پیشنهادی در بخش ۳ ارائه شده‌اند. همچنین، بخش ۴ به ارزیابی الگوریتم مدیریت دمای پیشنهادی و بررسی دقت مدل‌های دمایی پرداخته است. در نهایت، نتیجه‌گیری و پیشنهاداتی برای ادامه‌ی تحقیق در بخش ۵ بیان شده است.

## ۲- پیشینه

تا کنون در تحقیقات صورت گرفته در مدیریت پویای دمای پردازنده، از ابزار کنترلی دما مانند تنظیم فرکانس پردازنده، مهاجرت پردازه‌ها و تغییر درجه همروندی اجرای پردازه‌ها در یک زمانبند زمان اجرا برای دستیابی به اهداف استفاده شده است [۱]. به‌طور کلی، رویکرد حل مسئله در این تحقیقات را می‌توان در دو دسته‌ی واکنشی[۷] و پیشگیرانه[۸] تقسیم کرد (شکل ۲). در رویکرد واکنشی، پس از رسیدن دمای پردازنده به آستانه‌ی نامطلوب از ابزار کنترلی برای کاهش دما استفاده می‌شود. مشکل اساسی این روش این است که عدم‌تخطی از آستانه را تضمین نمی‌کند. علاوه بر این، نوسانات زیاد دمایی ایجاد شده در این روش، موجب کاهش عمرپردازنده می‌شود [۱۵]. در مقابل، رویکرد پیشگیرانه با بهره‌گیری از مدل‌های پیش‌بینی دما و انجام اقدامات کنترلی پیشگیرانه از رسیدن دما به محدوده‌ی نامطلوب جلوگیری می‌کند. مطالعات اخیر نشان داده است که رویکرد پیشگیرانه کارآمدی بیشتری در دستیابی به اهداف مدیریت دما از جمله عدم‌تخطی از آستانه‌ی دمای نامطلوب و اجتناب از کاهش کارایی برنامه دارد [۳].




شکل ۲- رویکردهای مدیریت دما

به طور کلی کارهای انجام شده در رویکرد پیشگیرانه‌ی دما را می‌توان در دو

دسته قرار داد. دسته‌ی اول، کارهایی هستند که در آن‌ها آگاهی از وجود برنامه‌های موازی مورد توجه قرار گرفته نشده است و مدیریت دما، هر یک از نخ‌های تولید شده توسط یک برنامه موازی را به عنوان برنامه‌ی مستقلی تلقی می‌کند [۱۳-۱۷]. براساس آزمایشات ما، عدم‌آگاهی از موازی‌سازی در برنامه‌ها می‌تواند هزینه‌های کارایی را در مدیریت دما افزایش دهد. دسته‌ی دوم از این کارها، آگاهی از وجود موازی‌سازی در برنامه‌ها را در نظر گرفته‌اند [۱۶-۲۵]. در این دسته، برخی از کارها به مسئله‌ی کنترل دما پرداخته [۱۶،۱۷، ۱۹] و برخی دیگر نیز در جهت بهبود توان و انرژی مصرفی پردازنده تلاش کرده‌اند [۲۰-۲۵]. گرچه بهبود توان مصرفی و یا انرژی ممکن است در راستای اهداف این پژوهش یعنی عدم‌تخطی از محدودیت دمایی نباشد، اما ایده‌های به کار رفته در این کارها ممکن است برای کنترل دمای پردازنده هم کارایی داشته باشند.

از میان تحقیقات اخیر، الگوریتم آگاه از همسایگی به‌عنوان یک رویکرد پیشگیرانه مورد توجه زیادی بوده است [۱۳]. در این الگوریتم، دمای هر کدام از هسته‌های پردازنده با در نظر گرفتن تاثیر هسته‌های همسایه به‌صورت جداگانه پیش‌بینی می‌شود. در این کار، چنانچه دمای پیش‌بینی شده برای یک هسته از محدودیت دمایی بیشتر باشد، هسته‌ای به عنوان مقصد جهت مهاجرت پردازهای هسته‌ی جاری انتخاب می‌شود. برای انتخاب هسته‌ی مقصد از پایین‌ترین دمای پیش‌بینی شده‌ی هسته‌ها استفاده می‌شود که از محدودیت مشخص شده پایین‌تر است. شکل ۳، نمودار گردشی الگوریتم آگاه از همسایگی را نمایش می‌دهد. در این الگوریتم، $T_i^{prev}$، $T_i^{cur}$ و $T_i^{pred}$، به ترتیب دمای گذشته، دمای حال و دمای پیش‌بینی شده برای هسته‌ی $i$ هستند و $T^{thr}$، $\Delta(t)$ و $S_i^{safe}$، به‌ترتیب، آستانه محدودیت دمایی، فاصله‌ی زمانی بین دو اجرای متوالی الگوریتم و فرکانس مطمئن برای هسته‌ی $i$، می‌باشند. همچنین، $H(P_k)$، فاکتور گرمایی هسته‌ی مقصد می‌باشد که از فرمول زیر بدست می‌آید:

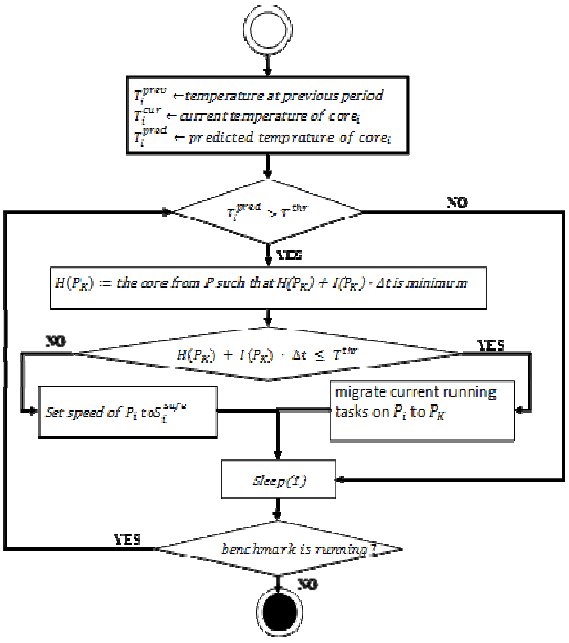
$$H(P_k) = \frac{\sum P_j \in P_k^{NB} \cup \{P_k\}^{T_j}}{|P_k^{NB} \cup \{P_k\}|} \quad (۱)$$

در واقع فرمول (۱)، میانگین دمای هسته‌ی $k$ و هسته‌های مجاور آن را محاسبه می‌کند. از دید الگوریتم آگاه از همسایگی، هسته‌ای که کمترین مقدار $H(P_k)$ را دارد برای انتخاب به عنوان مقصد مناسب است. افزون بر این، فاکتور $I(P_k)$ که بیانگر میزان شیب افزایش دمای هسته‌ی $k$ با درنظر گرفتن هسته‌های همسایه است، در الگوریتم آگاه از همسایگی از معیارهای انتخاب مقصد محسوب می‌شودکه مقادیر کمتر برای آن مطلوب‌تر هستند. $I(P_k)$ را می‌توان از فرمول زیر محاسبه کرد:

$$I(P_k) = \frac{\sum P_j \in P_k^{NB} \cup \{P_k\} \frac{T_j^{cur} - T_j^{prev}}{t_j^{cur} - t_j^{prev}}}{|P_k^{NB} \cup \{P_k\}|} \quad (۲)$$

در الگوریتم آگاه از همسایگی، چنانچه $H(P_k) + I(P_k).\Delta(t)$ برای هیچ یک از هسته‌ها پایین‌تر از محدودیت دمایی نباشد، فرکانس هسته‌ی جاری به سطح ایمن ($S_i^{safe}$) کاهش خواهد یافت تا دمای هسته‌ی جاری در آینده از محدودیت تعیین شده تجاوز نکند.

مزیت الگوریتم آگاه از همسایگی نسبت به دیگران، درنظرگرفتن تاثیر هسته‌های مجاور در مدل دمایی جهت پیش‌بینی دمای آینده برای یک هسته است تا تمایز میان دمای هسته‌ها مبنای مهاجرت پردازه‌ها قرار گیرد. به نظر می‌رسد، این روش در اجرای همزمان چندین برنامه‌ی غیرموازی کارایی مناسبی داشته باشد. در مقابل، در اجرای برنامه موازی ممکن است تفاوت چندانی میان فعالیت هسته‌های مختلف نباشد و مهاجرت نخ‌ها موجب کاهش دما نشود. در چنین حالتی کاهش شدید فرکانس در دستور کار قرار خواهد گرفت که بدون شک هزینه کارایی سنگینی را به مدیریت دما تحمیل خواهد کرد.




شکل ۳- نمودار گردشی الگوریتم آگاه از همسایگی [۱۳]

در [۱۴] برای پیش‌بینی دما علاوه بر دما و ویژگی فیزیکی هسته‌ها، ویژگی‌های بار کاری[۹] نیز درنظر گرفته شده است. در این روش یک مدل دمایی برای پیش‌بینی دما براساس بار کاری و یک مدل دمایی دیگر برای پیش‌بینی دما براساس ساختار پردازنده پیشنهاد شده است. در نهایت پیش‌بینی نهایی براساس مجموع ضرایبی از این دو، شکل می‌گیرد. این ضرایب براساس مشاهدات پیشین و به صورت تجربی بدست می‌آیند. زمانی که دمای یک هسته از یک مقدار مشخص بیشتر شود، هسته‌ای که کمترین پیش‌بینی دما را در آینده خواهد داشت، به عنوان مقصد مهاجرت انتخاب می‌شود.

در برخی از کارها از شمارنده‌های کارایی[۱۰] جهت پیش‌بینی دمای آینده استفاده شده است. در [۱۶]، با استفاده از یک آزمایش تجربی مشاهده شده که ترتیب مختلف اجرای وظایف در دما موثر است. به‌عنوان مثال اگر ابتدا یک وظیفه گرم و سپس یک وظیفه سرد اجرا شود، دمای هسته نسبت به حالت عکس کمتر خواهد بود. این مقاله به کمک شمارنده‌های کارایی یک تخمین زننده‌ی دما ارائه داده است که از آن در سیاستی استفاده شده که وظایف را به ترتیب دما مرتب کرده و داغ‌ترین وظیفه‌ای که دمای پردازنده را از حد آستانه بالاتر نمی‌برد را به عنوان وظیفه بعدی جهت اجرا انتخاب می‌کند.

یک پردازنده از واحدهای محاسباتی متفاوتی تشکیل شده است و یک برنامه و یا وظیفه ممکن است از تمامی این واحدهای محاسباتی استفاده نکند. و هر یک از وظایف ممکن است از واحدهای محاسباتی متفاوتی در پردازنده استفاده کنند. به عنوان مثال، یک وظیفه که عمده‌ی دستوراتش، اعمال ریاضی صحیح هستند، ALU را تحت تاثیر قرار می‌دهند در حالی که وظیفه‌ای که تمامی دستوراتش ممیز شناور است، واحد ممیز شناور را تحت تاثیر قرار داده و موجب افزایش دما در آن واحد می‌شود. در [۱۷]، سعی شده است تا در یک محیط چند نخی همزمان[۱۱]، وظایفی که از واحدهای محاسباتی متفاوت استفاده می‌کنند پشت سر هم اجرا شوند و یا وظایفی که از این نظر متمایز هستند همزمان اجرا شوند.

در [۱۸، ۱۹] یک الگوریتم توازن بار[۱۲] آگاه از دما ارائه شده است. که با استفاده از تغییر فرکانس و مهاجرت سعی می‌کند دما را زیر حد آستانه حفظ کند.

این روش در محیط برنامه‌نویسی charm++ پیاده‌سازی شده است. یکی از معایب مهم این کار، نیاز به آگاهی از زمان اجرای وظایف در الگوریتم زمانبندی است. باید توجه شود که در برنامه‌ها چنین اطلاعاتی به راحتی در دسترس نیست و به دست آوردن آن‌ها نیازمند پایش‌های آماری از زمان اجرا و استفاده از مدل‌های تخمینی است. در مقابل، از آنجایی که سیستم عامل از وجود وظایف موجود در برنامه مطلع نیست و در مدیریت اجرای آن‌ها نقشی ندارد، استفاده از این ایده برای مدیریت دما در سطح سیستم عامل، در عمل ممکن نیست.

یک برنامه‌ی موازی معمولاً علاوه بر قسمت‌هایی که موازی اجرا می‌شوند دارای قسمت‌هایی است که در آن موازی‌سازی صورت نگرفته و اجرا به صورت سریال است. بر این اساس، در برخی از کارها سعی شده است تا قسمت‌های سریال برنامه روی یک هسته‌ی قوی از پردازنده و قسمت‌های موازی برنامه روی دیگر هسته‌های ضعیف‌تر اجرا شوند تا ضمن افزایش کارایی برنامه، در مصرف انرژی صرفه‌جویی صورت گیرد. [۲۰، ۲۱]، مصرف انرژی در قسمت‌های سریال و موازی برنامه را با رویکردی در سطح کامپایلر کاهش داده‌اند. در اینکارها، در زمان کامپایل، قطعه کدی به قسمت‌های سریال برنامه افزوده می‌شود که فرکانس ساعت پردازنده را در حداکثر قرار می‌دهد و در قسمت‌های موازی برنامه قطعه کدی قرار داده می‌شود که سطح پایینی از فرکانس پردازنده را درخواست می‌کند. بنابراین، همواره قسمت‌های سریال برنامه با بیشترین فرکانس و قسمت‌های موازی با فرکانس پایین اجرا شده و در توان مصرفی صرفه‌جویی خواهد شد.

در تعداد زیادی از کارهای انجام شده در زمینه کاهش مصرف انرژی، تلاش شده است تا بهترین پیکربندی پردازنده برای اجرای برنامه پیش‌بینی شود تا ضمن کمترین تخریب کارایی، بیشترین صرفه‌جویی در مصرف انرژی صورت گیرد. به عبارت دیگر، در زمان اجرا و یا قبل از آن، مصرف انرژی و میزان کارایی برنامه در ترکیب‌های مختلفی از تعداد هسته‌ها و فرکانس ساعت پردازنده پیش‌بینی می‌شود و هر پیش‌بینی به عنوان یک نقطه‌ی عملیاتی[۱۳] سیستم تلقی می‌گردد. سپس، یکی از حالات که به اهداف موردنظر کاربر نزدیک‌تر است از میان نقاط عملیاتی انتخاب می‌شود.

در این راستا، [۲۲، ۲۳] از یک روش رگرسیون خطی[۱۴] برای دسته‌بندی و پیدا کردن نقاط عملیاتی سیستم استفاده کرده‌اند. در این کارها، داده‌های ورودی رگرسیون می‌تواند دما، شمارنده‌های کارایی، زمان اجرای برنامه، تعداد هسته‌ها و فرکانس پردازنده باشد. همچنین، خروجی زمان اجرا و انرژی مصرفی است. در این کارها، با افزایش تعداد انتخاب‌ها از میان نقاط عملیاتی ممکن، فضای جستجو بزرگ می‌شود و بررسی تمام این نقاط سربار زمان اجرا زیادی خواهد داشت. در [۲۴]، با استفاده از الگوریتم تپه‌نوردی[۱۵] و جستجوی دودویی فضای جستجوی نقاط عملیاتی هرس شده و بنابراین بار محاسباتی انتخاب نقطه‌ی عملیاتی بهینه کاهش می‌یابد گرچه ممکن است لزوماً بهترین حالت انتخاب نشود.

در اکثر کارهای انجام شده، زمانی که اهداف مختلفی از جمله کاهش دما، مصرف انرژی و زمان اجرای برنامه به صورت توامان موردنظر است، از روش بهینه‌سازی مبتنی بر محدودیت برای انتخاب تصمیم زمانبندی بهینه استفاده می‌شود. در این روش، یک هدف بیشینه یا کمینه می‌شود و دیگر اهداف به صورت محدودیت‌هایی در نظر گرفته می‌شوند.

به عنوان مثال، برای کمینه کردن زمان اجرا تلاش می‌شود که دما از محدودیت مشخصی فراتر نرود. با این حال، در معدودی از کارها سعی شده است تا تمامی اهداف به صورت همزمان بیشینه یا کمینه شوند. از این دست، در [۲۵] با استفاده از الگوریتم ژنتیک بهینه‌سازی توام کارایی، دما و توان مصرفی صورت گرفته است. با این وجود، سربار بالای زمان اجرای محاسبات تکاملی مانع کارآمدی راه حل پیشنهادی در این مقاله شده است.

## ۳- سیاست پیشنهادی مدیریت دما

سیاست پیشنهادی مدیریت پویای دما در این تحقیق بر مبنای یک مدل دمایی برای پیش‌بینی دمای پردازنده، یک مدل کارایی جهت پیش‌بینی میزان افزایش سرعت و یک الگوریتم انتخاب پیکربندی است که به کمک این دو مدل ضمن تضمین رعایت محدودیت دما، سعی در کاهش تخریب کارایی دارد. پس از انجام آزمایشات مختلف بر روی برنامه‌های محک موازی، پی‌بردیم که با تغییر تعداد نخ‌های فعال برنامه و فرکانس پردازنده، زمان اجرای برنامه تقریباً به صورت خطی و دمای پردازنده به صورت غیرخطی تغییر می‌کنند. بنابراین، ما با پیشنهاد دو مدل کارایی و دمایی، نقش عواملی همچون فرکانس ساعت پردازنده و تعداد هسته‌های فعال را در کارایی برنامه و دمای پردازنده تبیین کرده و الگوریتم مدیریت دمایی خود را بر پایه این مدل‌ها پیشنهاد می‌دهیم.

### ۳-۱- مدل پیش‌بینی دما

یک هدف اساسی در مدل پیش‌بینی دما، افزایش دقت در مقادیر پیش‌بینی شده است. از این رو، برای افزایش دقت پیش‌بینی مدل دما بایستی ویژگی‌های موثر در دمای پردازنده شناسایی و در مدل گنجانده شوند. با این وجود، شناخت این ویژگی‌ها نیازمند صرف آزمایشات متعدد و آزمودن ترکیبات مختلفی از ویژگی‌های آماری دمای پردازنده است. گرچه روش‌های متعددی برای ایجاد مدل‌های دمایی با استفاده از ویژگی‌های آماری موجود است، در این تحقیق ما از رگرسیون خطی برای تخمین ضرایب ویژگی‌های موثر مدل دمایی خود بهره بردیم که در تحقیقات دیگر مرسوم بوده است [۱۴]. فرمول (۳) یک معادله درجه اول از مدل دمایی پیشنهادی را نشان می‌دهد:

$$T_{new} = \alpha_1 T_{cur} + \alpha_2 f_{new} + \alpha_3 c_{new} + \alpha_4 \qquad (۳)$$

که در آن $T_{new}$ پیش‌بینی دمای جدید، $T_{cur}$ دمای حال، $f_{new}$ فرکانس آینده پردازنده، $c_{new}$ تعداد نخ‌های کارگر آینده و مقادیر $\alpha_i$ برای $i = \{1,2,\dots,4\}$ ضرایب تعیین شده توسط رگرسیون خطی هستند. پس از انجام آزمایشات متعدد روی برنامه‌های محک مختلف، به این نتیجه رسیدیم که علاوه بر ویژگی‌های استفاده شده در فرمول (۳) عوامل دیگری چون شیب تغییرات ویژگی‌ها نیز بر عملکرد الگوریتم مدیریت دمایی موثر است. فرمول (۴)، با دخیل کردن پارامترهای بیشتر را در یک معادله‌ی درجه دوم برای مدل دمایی پیشنهاد می‌دهد که در آن علاوه بر ویژگی‌های فرمول (۴)، میزان تغییرات ویژگی‌ها نیز جهت افزایش دقت مدل، گنجانده شده است.

$$\begin{aligned} T_{new} = {} & \alpha_1 T_{cur} + \alpha_2 f_{new} + \alpha_3 c_{new} + \alpha_4 \nabla T_{cur} + \alpha_5 \nabla f_{new} + \\ & \alpha_6 \nabla c_{new} + \alpha_7 T_{cur}^2 + \alpha_8 f_{new}^2 + \alpha_9 c_{new}^2 + \alpha_{10} \nabla T_{cur}^2 + \\ & \alpha_{11} \nabla f_{new}^2 + \alpha_{12} \nabla c_{new}^2 + \alpha_{13} \end{aligned} \qquad (۴)$$

عملگر گرادیان ($\nabla$) برای یک ویژگی به معنی تفاضل مقدار جدید آن ویژگی با مقدار قبلی آن در دو نمونه‌ی متوالی از زمان اجرا است. به عبارت دیگر برای ویژگی فرضی $\chi$ خواهیم داشت:

$$\nabla\chi_{cur} = \chi_{cur} - \chi_{past} \quad , \quad \nabla\chi_{new} = \chi_{new} - \chi_{cur}$$

که در آن، $\chi_{past}$، $\chi_{cur}$ و $\nabla\chi_{new}$ به ترتیب مقدار پیشین، حال و آینده‌ی ویژگی فرضی $\chi$ است.

## ۳-۲- مدل پیش‌بینی کارایی

در مدل پیش‌بینی کارایی ما به دنبال تخمین تغییر در زمان اجرای برنامه در اثر تغییر پیکربندی سیستم هستیم. از این رو، تخمین توان پردازشی سیستم را به صورت زیر تبیین می‌کنیم:

$$\text{ProcessingPower} = \text{f} \times \text{c}$$

که در آن f فرکانس ساعت و c تعداد هسته‌های فعال پردازنده است.

در بسیاری از الگوریتم‌های مدیریت دمای پردازنده، از تنظیم در فرکانس ساعت و تغییر در تعداد هسته‌های فعال در پردازنده استفاده می‌شود. بنابراین، میزان تغییر در کارایی برنامه را پس از اعمال تغییرات نسبت به حالت کنونی را می‌توان از تقسیم تخمین توان پردازشی کنونی بر تخمین توان پردازشی جدید محاسبه کرد که خواهیم داشت:

$$\text{Speedup} = \frac{\text{f}_{\text{new}} \times \text{c}_{\text{new}}}{\text{f}_{\text{cur}} \times \text{c}_{\text{cur}}} \quad (۵)$$

که در آن $\text{f}_{\text{cur}}$ و $\text{c}_{\text{cur}}$ به ترتیب فرکانس فعلی و تعداد هسته‌های فعال فعلی پردازنده است. همچنین، $\text{f}_{\text{new}}$ و $\text{c}_{\text{new}}$ به‌ترتیب فرکانس آینده و تعداد هسته‌های فعال آینده پردازنده است.

بدون شک، هر چه مقدار Speedup در فرمول (۵) بزرگتر باشد، افزایش سرعت بیشتری در اثر تغییر در فرکانس ساعت و تعداد هسته‌های فعال پردازنده حاصل خواهد شد. بنابراین، افزایش مقدار Speedup می‌تواند از اهداف الگوریتم مدیریت دما باشد.

## ۳-۳- الگوریتم مدیریت دما

ما در الگوریتم پویای دمای پیشنهادی، به نحوی تعداد هسته‌های فعال و فرکانس ساعت پردازنده‌ی چندهسته‌ای را تعیین می‌کنیم که ضمن تضمین عدم‌تخطی دما از میزان آستانه، حداقل آسیب کارایی به برنامه وارد شود. در شکل ۴، الگوریتم مدیریت دمای پیشنهادی نمایش داده شده است. الگوریتم پیشنهادی ما شامل دو بخش مقداردهی اولیه و اجرای متوالی است. در بخش مقداردهی اولیه که در خطوط یک تا پنج در شکل ۴ مشاهده می‌شود، مجموعه‌ی فرکانس‌های ساعت و تعداد هسته‌های پردازنده در متغیرهایی ذخیره شده و نیز تعداد هسته‌های فعال برابر با تمام هسته‌ها و فرکانس ساعت برابر با حداکثر فرکانس ممکن مقداردهی می‌شوند. همچنین، محدودیت دمایی کاربر به آستانه‌ی دمایی ایمن تبدیل می‌شود. آستانه دمایی ایمن، عدم‌تخطی از محدودیت دمایی کاربر را تضمین می‌کند و همواره مقداری کمتر یا مساوی آن است. برای بدست آوردن مقدار آستانه دمایی ایمن، می‌توان ابتدا در الگوریتم مدیریت دمای پیشنهادی مقدار آستانه را برابر با محدودیت دمایی کاربر فرض کرد. سپس، با اجرای برنامه‌های محک مختلف مقدار حداکثر دمای اندازه‌گیری شده برای هر آستانه را می‌توان به عنوان محدودیت دمایی قابل تضمین با آن آستانه محسوب کرد. به عنوان مثال، ممکن است نیاز باشد برای تضمین عدم‌تخطی از محدودیت دمایی ۶۰ درجه، میزان آستانه دمایی ایمن را برابر با ۴۸ درجه در نظر گرفت.

در بخش دوم که خطوط شش تا ۱۴ الگوریتم را شامل می‌شود، فرآیندی توصیف می‌شود که در طی مدت اجرای برنامه‌ی موازی در سیستم ادامه می‌یابد. در این فرآیند، ابتدا (خط شش) تنظیمات جدید فرکانس و تعداد هسته‌های فعال پردازنده با محاسباتی تعیین می‌شود که در آن بیشینه‌ی کارایی با شرط عدم‌تجاوز از آستانه ایمن دما حاصل می‌آید. برای انجام چنین محاسباتی از بین تمامی ترکیبات تعداد هسته‌ها و فرکانس پردازنده که در آن‌ها دمای پردازنده با استفاده از فرمول (۳) یا (۴) کمتر از مقدار آستانه تخمین زده می‌شود، گزینه‌ای انتخاب شود که براساس فرمول (۵) بیشترین کارایی را دارد. در گام بعدی، اگر تعداد هسته‌های فعال جدید کمتر از هسته‌های فعال کنونی باشند، به تعداد تفاضل این دو، از نخ‌های کارگر فعال موجود معلق خواهند شد تا در عمل از هسته‌های فعال پردازنده کاسته شود. همچنین، اگر تعداد هسته‌های فعال جدید بیشتر از هسته‌های فعال کنونی باشد، به تعداد تفاضل این دو از نخ‌های کارگری که قبلا تعلیق شده‌اند برای ادامه‌ی اجرا بیدار خواهند شد. در نهایت، فرکانس پردازنده با فرکانس جدید تنظیم خواهد شد. لازم به ذکر است که در عموم پیاده‌سازی‌های رابایش کار، تعداد نخ‌های کارگر در ابتدای اجرا برابر با تعداد هسته‌های پردازنده است.

```
1: Φ ← set of processor frequencies
2: P ← set of processor cores
3: f_cur ← maximum processor frequency
4: c_cur ← the number of processor cores
5: T_safe ← getSafeThreshold(T_constraint )
6: while(a parallel program is running) do
7:   f_new , c_new ← argmax_{f_new ∈Φ, c_new ∈P} {Speedup | T_new ≤ T_safe }
8:   if c_new ≤ c_cur then
9:     Suspend (c_new − c_cur ) worker threads
10:  else
11:    Awake (c_cur − c_new ) worker threads
12:  end if
13:  f_cur ← f_new
14:  c_cur ← c_new
15:  sleep(1)
16: end while
```

شکل ۴- الگوریتم مدیریت دمای پیشنهادی

# ۴- ارزیابی

ما الگوریتم مدیریت دمای پیشنهادی خود را بر روی یک سیستم واقعی دارای پردازنده‌ی Intel Core i7-4790K که دارای چهار هسته‌ی فیزیکی، هشت هسته منطقی و ۱۶ پله‌ی فرکانسی مختلف است آزمایش کردیم. این سیستم دارای ۸ گیگابایت حافظه اصلی بود و از سیستم عامل Ubuntu 14.04 به عنوان میزبان استفاده شد. برای پیاده‌سازی الگوریتم پیشنهادی از زبان C++ و کامپایلر GCC استفاده شد. همچنین، برنامه‌های محک مورد آزمایش شامل مرتب‌سازی ادغامی، ضرب استاندارد ماتریس، تبدیل فوریه سریع، ترانهاده ماتریس، مرتب‌سازی سریع و مرتب‌سازی رقمی با استفاده از ابزار Intel Cilk Plus [۲۸] پیاده‌سازی شدند. افزون بر این، ما برای خواندن دمای هسته‌های پردازنده از ابزار lm-sensors [۲۶] و برای تنظیم فرکانس پردازنده از ابزار CPUFreq [۲۷] استفاده کردیم. در طول زمان آزمایشات تلاش شد تا دمای اتاق ثابت و برای تمامی آزمایشات یکسان نگاه داشته شود. همچنین، ما با استفاده از ابزار lm-sensors سرعت خنک‌کننده‌ی پردازنده را در تمامی آزمایشات ثابت نگاه داشتیم تا تغییر رفتار خنک‌کننده در نتایج اثری نداشته باشد.

## ۴-۱- ارزیابی دقت مدل دمایی

ما هر یک از مدل‌های دمایی ارئه شده در بخش ۱-۳ را در الگوریتم مدیریت دمایی پیشنهادی قرار داده تا اثر افزایش دقت مدل‌ها بر عملکرد سیاست پیشنهادی مدیریت دما را ارزیابی کنیم. در هر اجرا، دمای پردازنده به صورت لحظه‌ای اندازه‌گیری شد و با دمای پیش‌بینی شده برای آن لحظه مقایسه گردید. شکل ۵، نتایج حاصل از این آزمایش را برای برنامه‌های محک مختلف نشان

می‌دهد. در نمودارهای ارائه شده در شکل ۵، محور افقی محدودیت‌های دمایی مختلف و محور عمودی میزان خطای مطلق پیش‌بینی دمای پردازنده برحسب درجه سانتیگراد است. همان‌طور که در این شکل مشاهده می‌شود، برای بیشتر برنامه‌های محک، مدل دمایی فرمول (۳) از نظر میزان قدر مطلق خطای پیش‌بینی، عملکرد بهتری نسبت به مدل دمایی فرمول (۲) دارد. به طور میانگین، مدل دمایی فرمول (۳) از نظر زمان اجرا به میزان ۴۴ درصد و از نظر میانگین خطای پیش‌بینی ۱۶ درصد عملکرد بهتری نسبت به مدل دمایی فرمول (۲) دارد. این نتایج نشان می‌دهد که افزایش دقت مدل به میزان قابل توجهی در کارایی برنامه و عملکرد الگوریتم مدیریت دمایی موثر است. بنابراین، ما مدل پیش‌بینی دمای فرمول (۳) را برای استفاده در سیاست پیشنهادی مدیریت دمای خود انتخاب کردیم.

## ۴-۲- آستانه دمایی ایمن

همان‌طور که در بخش ۳-۳ بیان شد، در ابتدای اجرای الگوریتم پیشنهادی مدیریت دما نیاز است تا محدودیت دمایی درخواستی کاربر به آستانه دمایی ایمن تبدیل شود. برای این منظور، ما آستانه‌های دمایی ایمن را با اجرای برنامه‌های محک بر روی بستر آزمایش و براساس توضیحات بخش ۳-۳ محاسبه کردیم. جدول ۱، مقادیر آستانه محاسبه شده را برای تعدادی از محدودیت‌های دمایی کاربر نمایش می‌دهد. همان‌طور که در این جدول مشاهده می‌شود، همواره آستانه دمایی ایمن برای هر محدودیت، مقداری کمتر از آن است.

## ۴-۳- ارزیابی روش پیشنهادی

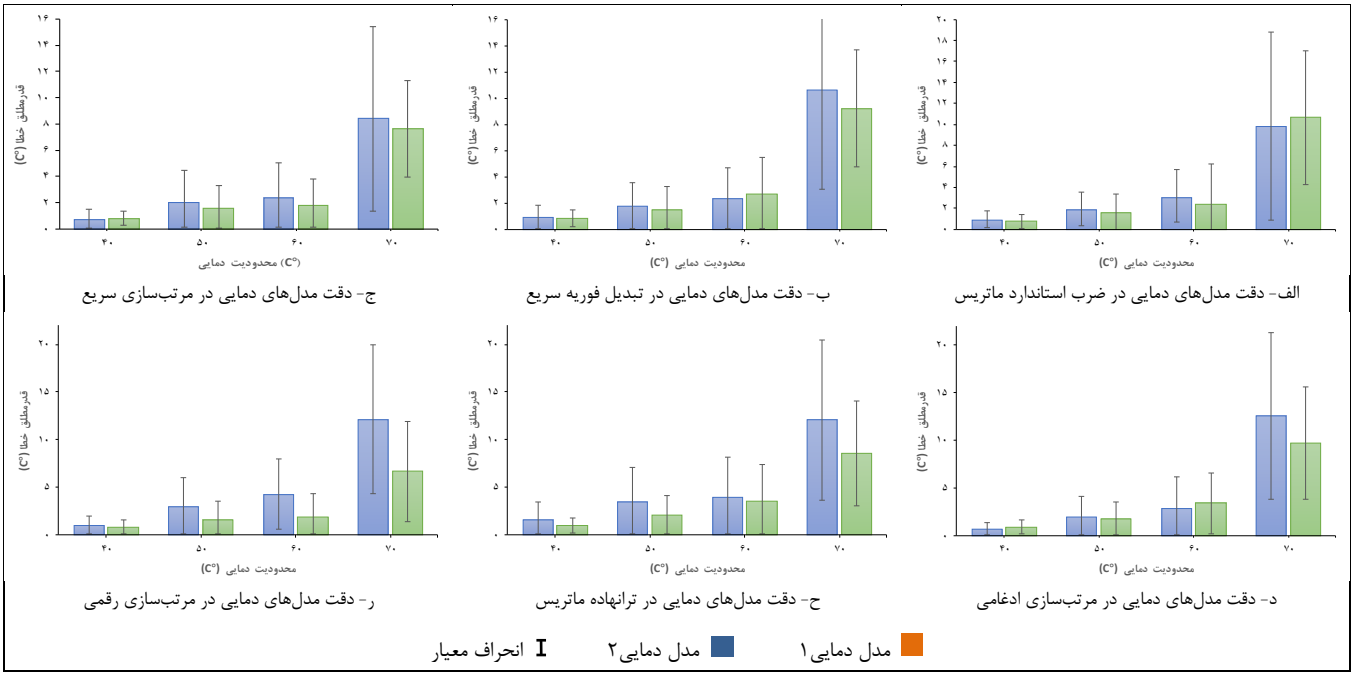
برای ارزیابی الگوریتم پیشنهادی، ما الگوریتم مدیریت پویای دمای آگاه از همسایگی [۴] را پیاده‌سازی کردیم. سپس، در اجرای آزمایشات دمای پردازنده را به صورت لحظه‌ای نمونه‌گیری کرده و مدت زمان اجرای برنامه‌های محک ثبت نمودیم. محدودیت‌های دمایی در نظر گرفته شده در آزمایشات شامل ۴۰، ۵۰، ۶۰ و ۷۰ درجه سانتیگراد بودند.

جدول ۱- آستانه دمایی ایمن برای محدودیت کاربر

| محدودیت دمایی کاربر | آستانه دمایی ایمن |
|---|---|
| ۴۰ | ۳۵ |
| ۴۵ | ۳۸ |
| ۵۰ | ۴۰ |
| ۵۵ | ۴۱ |
| ۶۰ | ۴۷ |
| ۶۵ | ۵۰ |
| ۷۰ | ۵۵ |

نمودارهای ارائه شده در شکل ۶، نتایج حاصل از این آزمایشات را به‌صورت خلاصه بیان کرده است. در هر یک از این نمودارها، محور افقی زمان اجرای برنامه محک بر حسب ثانیه و محور عمودی دمای پردازنده بر حسب درجه سانتیگراد است. براساس این نتایج، در تمامی موارد زمان اجرای برنامه محک تحت مدیریت الگوریتم پیشنهادی بسیار کمتر از زمان اجرا تحت مدیریت الگوریتم آگاه از همسایگی است. همچنین، الگوریتم پیشنهادی تمامی محدودیت‌های دمایی را رعایت کرده و هرگز اجازه نمی‌دهد که دمای پردازنده از محدودیت تعیین شده فراتر رود. این در حالی است که الگوریتم آگاه از همسایگی به‌طور میانگین در ۲۹ درصد از زمان اجرا از محدودیت دمایی تعیین شده تخطی کرده و نوسانات شدیدی در دمای پردازنده ایجاد می‌کند.

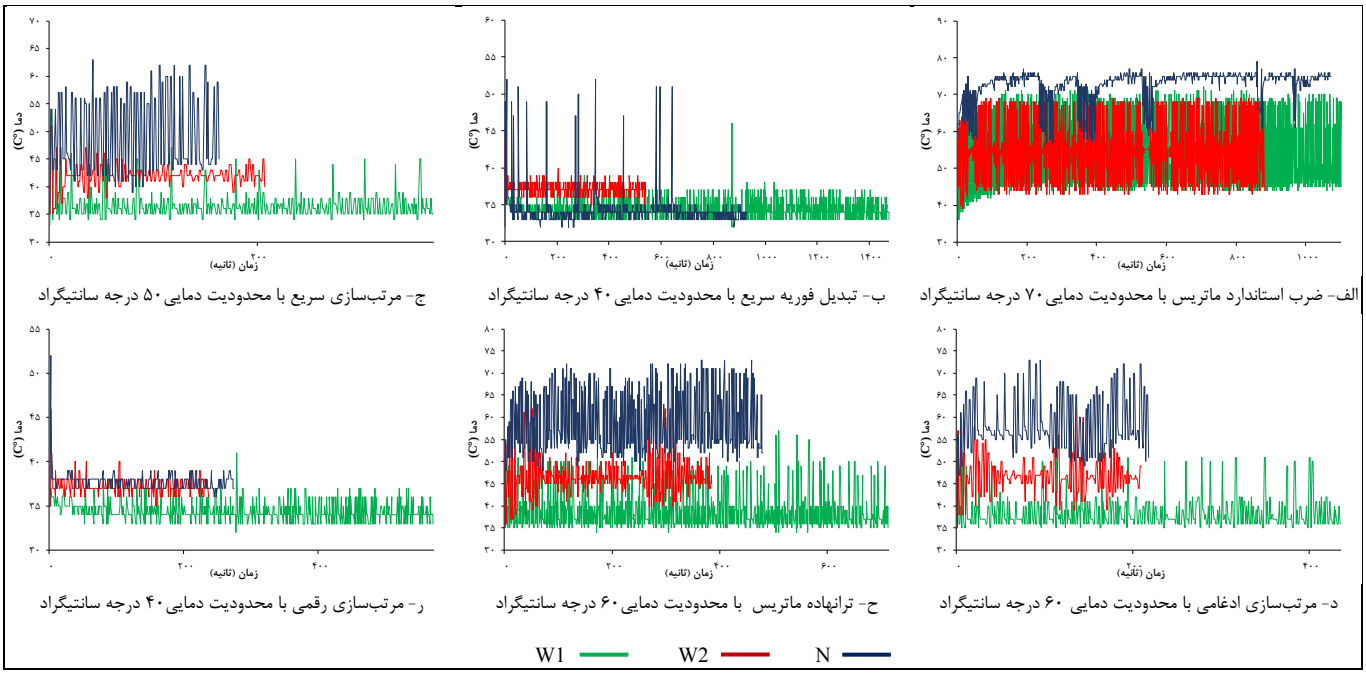
شکل ۷، میانگین زمان اجرای برنامه‌های محک مختلف در محدودیت‌های دمایی ۴۰، ۵۰، ۶۰ و ۷۰ درجه را نمایش می‌دهد. در این شکل، ستون مربوط به میانگین زمان اجرا در الگوریتم پیشنهادی با W1 و W2، و برای الگوریتم آگاه از همسایگی با حرف N نمایش داده شده است. همچنین، از حرف P برای نمایش زمان اجرا توسط زمانبند performance استفاده شده است. در هر ستون، ارتفاع آبی رنگ مدت زمانی است که دما در محدوده‌ای کمتر از محدودیت تعیین قرار داشته است. همچنین، میانگین زمان تخطی از محدودیت دمایی با رنگ نارنجی نمایش داده شده است.



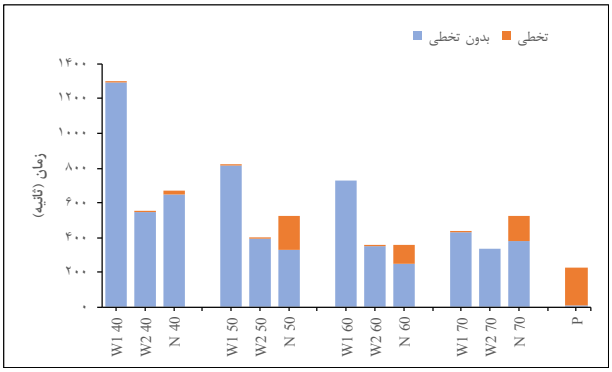

شکل۵- دقت مدل‌های دمایی در برنامه‌های محک مختلف. محور افقی آستانه‌های ایمن دمایی مختلف و محور عمودی قدرمطلق خطای پیش‌بینی دما

بر این اساس، مشاهده می‌گردد که توسط الگوریتم آگاه از همسایگی در محدودیت‌های دمایی مختلف به طور میانگین برای ۱۵ درصد از زمان اجرا، دمای پردازنده از محدودیت تعیین شده تخطی کرده است. این در حالی است که الگوریتم پیشنهادی تقریباً در تمامی موارد توانسته است دمای پردازنده را کاملا زیر محدودیت تعیین شده مدیریت کند و به‌طور میانگین ۲۸ درصد نسبت به روش آگاه از همسایگی و ۴۶ درصد نسبت به روش پایه با مدل دمایی یک، کارایی بهتری داشته باشد.

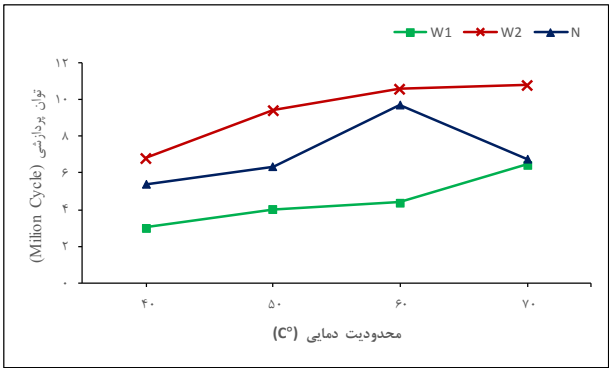
همچنین، ما مشاهده کردیم که همواره میانگین توان پردازشی در الگوریتم پیشنهادی نسبت به الگوریتم آگاه از همسایگی بیشتر بوده ولی این توان بیشتر با ترکیب تعداد هسته‌های فعال بیشتر و فرکانس پایین‌تر حاصل آمده است. شکل ۸، میانگین توان پردازشی پردازنده را برای الگوریتم پیشنهادی و الگوریتم آگاه از همسایگی در اجرای برنامه‌های محک نمایش داده است. با توجه به این شکل، هر چه سطح محدودیت دما بیشتر باشد، الگوریتم‌های مدیریت دما ناگزیر به اعمال سطح پایین‌تری از توان پردازشی هستند که منجر به افزایش زمان اجرای برنامه‌های موازی خواهد شد.




شکل ۶- مقایسه عملکرد روش‌ها در برنامه‌های محک مختلف. محور افقی زمان برحسب ثانیه و محور عمودی دمای لحظه‌ای برحسب درجه سانتیگراد




شکل۷- میانگین زمان اجرای برنامه‌های محک تحت محدودیت‌های دمایی ۴۰، ۵۰، ۶۰ و ۷۰ درجه سانتیگراد




شکل۸- میزان توان پردازشی الگوریتم پیشنهادی در مقایسه با الگوریتم آگاه از همسایگی

## ۵- نتیجه‌گیری

در این پژوهش، ما یک الگوریتم مدیریت پویای دما را در سطح سیستم عامل پیشنهاد دادیم که دمای پردازنده‌ی چندهسته‌ای را در اجرای برنامه‌های موازی پیاده‌سازی شده توسط زمانبند رایش کار مدیریت می‌کند. الگوریتم ما مبتنی بر دو مدل دمایی و کارایی پیشنهادی است که دمای پردازنده و تغییرات زمان اجرای را پیش‌بینی می‌کنند. همچنین تاثیر افزایش دقت مدل دمایی را نیز مورد بررسی قرار دادیم و مشاهده کردیم که افزایش دقت مدل نه‌تنها زمان اجرای برنامه را کاهش می‌دهد بلکه برعملکرد الگوریتم مدیریت دما نیز موثر بوده و خطای پیش‌بینی دما را نیز کاهش می‌دهد. آزمایشات بر روی سیستم واقعی نشان داد که الگوریتم پیشنهادی ما عملکرد بسیار بهتری از الگوریتم آگاه از همسایگی دارد. بنابراین، ما الگوریتم پیشنهادی خود را راه حلی مناسب برای مدیریت دمای پردازنده‌ی چندهسته‌ای در اجرای برنامه‌های موازی رایش کار می‌دانیم.

در آینده تصمیم داریم دقت مدل‌های دمایی و کارایی را بهبود بخشیم و اثر این بهبود را در عملکرد الگوریتم مدیریت دمای پیشنهادی ارزیابی کنیم. همچنین، قصد داریم تا ارزیابی‌های خود را بر روی پردازنده‌ها و برنامه‌های محک متنوع‌تری گسترش دهیم.

## مراجع


[1] J. Kong, S. W. Chung, and K. Skadron, "Recent thermal management techniques for microprocessors,"In Computing Surveys (CSUR), vol. 44, no. 3, pp.1-42, 2012.

[2] J. Diaz, C. Munoz-Caro, and A. Nino, "A survey of parallel programming models and tools in the multi and many-core era," IEEE Transactions on parallel and distributed systems, vol. 23, no. 8, pp. 1369-1386, 2012.

[3] S. Zhuravlev, J. C.Saez, Blagodurov, S. Fedorova, and M. Prieto, "Survey of energy-cognizant scheduling techniques," IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 7, pp. 1447-1464, 2013.

[4] T. van Dijk, and C. Jaco, "Lace: non-blocking split deque for work-stealing," In European Conference on Parallel Processing, pp. 206-217, 2014.

[5] A. Morrison, and Y. Afek, "Fence-free work stealing on bounded TSO processors," ACM SIGPLAN, vol. 49, no. 4, pp. 413-426, 2014.

[6] J. Nakashim, S.Nakatani, and K. Taura, "Design and implementation of a customizable work stealing scheduler," In Proceedings of the 3rd International Workshop on Runtime and Operating Systems for Supercomputers, pp. 1-9, 2013.

[7] M. Wimme, D. Cederma, J. L. Träff, and P. Tsigas, "Work-stealing with configurable scheduling strategies," In ACM SIGPLAN, vol. 48, no. 8, pp. 315-316, 2013.

[8] A. Bhattacharjee, and M. Martonosi, "Thread criticality predictors for dynamic performance, power, and resource management in chip multiprocessors," InACM SIGARCH Computer Architecture, vol. 37, no. 3, pp. 290-301, 2009.

[9] D. Hendler, and N. Shavi, "Non-blocking steal-half work queues," In Proceedings of the twenty-first annual symposium on Principles of distributed computing, pp. 280-289, 2002.

[10] U. Acar, A. Charguéraud, and M. Rainey, "Scheduling parallel programs by work stealing with private deques," In ACM SIGPLAN, vol. 48, no. 8, pp. 219-228, 2013.

[11] D. Hendler, Y. Lev, M. Moir, and N. Shavit, "Adynamic-sized nonblocking work stealing deque," Technical report, Sun Microsystems, 2005.

[12] S. Imam, and V. Sarkar, "Load balancing prioritized tasks via work-stealing," In Euro-Par'15, pp. 222-234, 2015.

[13] G. Liu, M. Fan, G. Quan, and M. Qiu, "On-Line predictive thermal management under peak temperature constraints for practical multi-core platforms,"In Journal of Low Power Electronics, vol. 8, no. 5, pp. 565-578, 2012.

[14] I. Yeo, C. C. Liu, and E. J. Kim, "Predictive dynamic thermal management for multicore systems," In Proceedings of the 45th annual Design Automation Conference, pp. 734-739, 2008.

[15] R. Cochran, and R. Sherie, "Thermal prediction and adaptive control through workload phase detection," ACM Transactions on Design Automation of Electronic Systems, vol. 18, no. 7, 2013.

[16] J. Yang, X. Zhou, M. Chrobak, Y. Zhang, and L. Jin, "Dynamic thermal management through task scheduling," IEEE International Symposium on Performance Analysis of Systems and software (ISPASS'08), pp. 191-201, 2008.

[17] A. Merkel, and F. Bellosa, "Task activity vectors: a new metric for temperature-aware scheduling," In SIGOPS Operating Systems Review, vol. 42, no. 4, pp.1-12, 2008.

[18] O. Sarood, P. Miller, E. Totoni, and L.V. Kale, "Load Balancing for High Performance Computing Data Centers," IEEE Transactions on Computers, vol. 61, no. 12, pp. 1752-1764, 2012.

[19] O. Sarood, "Optimizing performance under thermal and power constraints for HPC data centers,"Doctoral dissertation, University of Illinois at Urbana-Champaign, 2014.

[20] T. Lu, P. P. Pande, and B. Shirazi,"A Dynamic, Compiler Guided DVFS Mechanism to Achieve Energy-Efficiency in Multi-coreProcessors," Sustainable Computing: Informatics and Systems, vol. 12, pp. 1-9, 2016.

[21] R. A. Shafik, A. Das, S. Yang, G. Merrett, and B. M. Al-Hashimi, "Adaptive energy minimization of OpenMP parallel applications on many-core systems," In Proceedings of the 6th Workshop on Parallel Programming and Run-Time Management Techniques for Many-core Architectures, pp. 19-24, 2015.

[22] R. Cochran, C. Hankendi, and A. Coskun, "Identifying the optimal energy-efficient operating points of parallel workloads," In IEEE/ACM International Conference on Computer-Aided Design, pp. 608-615, 2011.

[23] D. D. Sensi, "Predicting Performance and Power Consumption of Parallel Applications," International Conference on Parallel, Distributed, and Network-Based Processing, pp. 200-207, 2016.

[24] J. Li, and J. F. Martinez, "Dynamic power-performance adaptation of parallel computation on chip multiprocessors," In The Twelfth International Symposium on High-Performance Computer Architecture, pp. 77-87, 2006.

[25] H. F. Sheikh, I. Ahmad, and D. Fan, "An Evolutionary Technique for Performance-Energy-Temperature Optimized

Scheduling of Parallel Tasks on Multi-Core Processors," IEEE Transactions on Parallel and Distributed Systems, vol. 27, no. 3, pp. 668-681, 2016.

[26] lm-sensors Linux hardware monitoring [Online]. Available: http://www.lm-sensors.org, Jan 2017.

[27] Linux cpufreq governors, Linux Kernel [Online]. Available:https://www.kernel.org-/doc/Documentation/cpu-freq/governors.txt. Jan 2017.

[28] Intel Cilk Plus [Online]. Available: http://www.cilkplus.org, Jan 2017.

**حمید گوهرجو** دانش آموخته رشته علوم کامپیوتر دانشگاه گلستان- گرگان در مقطع کارشناسی. در حال حاظر دانشجوی کارشناسی‌ارشد رشته مهندسی کامپیوتر - نرم‌افزار دانشگاه فردوسی مشهد و عضو آزمایشگاه معماری کامپیوتر پیشرفته دانشگاه فردوسی مشهد. علاقه‌مندی‌ها: برنامه‌نویسی موازی، زمانبندی، سیستم عامل و مدیریت دمای پردازنده.
آدرس پست‌الکترونیکی ایشان عبارت است از:
ha.goharjoo@mail.um.ac.ir

**مرتضی مرادی** در سال ۱۳۸۶ از دانشگاه بیرجند در مقطع کارشناسی رشته مهندسی کامپیوتر فارغ‌التحصیل شد. سپس، در سال ۱۳۹۱ موفق به اخذ مدرک کارشناسی‌ارشد از دانشگاه آزاد اسلامی واحد مشهد شد. او در حال حاضر دانشجوی دکتری مهندسی کامپیوتر- نرم‌افزار گرایش سیستم‌های نرم‌افزاری در دانشگاه فردوسی مشهد است. علایق تحقیقاتی وی شامل طراحی الگوریتم‌های موازی، مدیریت حافظه‌های اشتراکی و توزیع شده، کنترل دما و توان مصرفی پردازنده و پرداختن به مسئله زمانبندی اجرای وظایف است.
آدرس پست‌الکترونیکی ایشان عبارت است از:
morteza.moradi@mail.um.ac.ir

**حمید نوری** مقطع کارشناسی و کارشناسی‌ارشد خود را به ترتیب در سال ۱۳۷۵ و ۱۳۷۹ در رشته مهندسی کامپیوتر از دانشگاه صنعتی شریف و دانشگاه صنعتی امیرکبیر دریافت کرد. مقطع دکتری خود را در رشته مهندسی کامپیوتر در دانشگاه کیوشو در ژاپن گذرانده است. در سال‌های ۱۳۸۷ تا ۱۳۸۹ استادیار دانشکده مهندسی برق و کامپیوتر دانشگاه تهران بوده است. از سال ۱۳۸۹ تا به اکنون استادیار گروه مهندسی کامپیوتر در دانشگاه فردوسی مشهد و سرپرست آزمایشگاه معماری کامپیوتر پیشرفته می‌باشد. زمینه‌های تحقیقاتی ایشان شامل معماری کامپیوتر، پردازنده‌های چند هسته‌ای، پردازش موازی و سیستم‌های نهفته می‌باشد.
آدرس پست‌الکترونیکی ایشان عبارت است از:
hnoori@um.ac.ir

**اطلاعات بررسی مقاله:**

تاریخ ارسال: ۱۳۹۵/۱۰/۲۷
تاریخ اصلاح: ۱۳۹۵/۱۱/۲۰
تاریخ قبول شدن: ۱۳۹۵/۱۱/۲۹
نویسنده مرتبط: دکتر حمید نوری، دانشکده مهندسی، دانشگاه فردوسی مشهد، مشهد، ایران.

[1]Work Stealing
[2]Predictive Dynamic Thermal Management (PDTM)
[3]Dynamic Voltage and Frequency Scaling (DVFS)
[4]Task Migration
[5]Worker Thread
[6]Hot Spots
[7]Reactive
[8]Predictive
[9]Workload
[10]Performance Counters
[11]Simultaneous Multi-Threading (SMT)
[12]Load Balancing
[13]Operating Points
[14]Leaner Regression
[15]Hill Climbing