
بسم الله الرحمن الرحیم

دانشکده مهندسی

گروه کامپی‌وتر

پایان‌نامه

جهت اخذ درجه کارشناسی ارشد

در رشته مهندسی کامپیوتر گرایش نرم‌افزار

عنوان:

پیش‌بینی بن‌بست در زمان اجرا برای برنامه‌های همروند با استفاده از شبکه‌های عصبی

استاد راهنما:

دکتر سید مرتضی بابامیر

استاد مشاور:

دکتر حسین ابراهیم‌پور کومله

به وسیله:

المیرا حسن‌زاده

شهریور ماه ۱۳۹۱

تقدیم به:

پدر و مادر عزیزم

با سپاس از:

استاد محترم جناب آقای دکتر سید مرتضی بابامیر

به‌واسطه رهنمودها و پشتیبانی بی‌دریغشان در به ثمر رسیدن این پایان‌نامه.

استاد محترم جناب آقای دکتر حسین ابراهیم‌پور کومله

به‌واسطه توصیه‌های مفیدشان در بهبود محتوای این پایان‌نامه.

و همچنین

از تشریک مساعی جناب آقای دکتر نوروزی و جناب آقای دکتر فرجی به‌عنوان اساتید داور که این پایان‌نامه را مورد مطالعه قرار دادند و در جلسه دفاعیه شرکت نمودند، تشکر می‌نمایم.

و در پایان

از جناب آقای دکتر ایمانیان که به‌عنوان نماینده تحصیلات تکمیلی دانشگاه قبول زحمت نموده‌اند نیز کمال تشکر و قدردانی را دارم.

چکی‌ده

افزایش روزافزان استفاده از برنامه‌نویسی همروند به دلیل بهبودهایی که در زمان پاسخ‌گویی، راندمان و استفاده بهینه از منابع مهیا می‌کند، در نرم‌افزارهای امروزی مشهود است. با وجود همه‌ی خواست‌گاهی که برای برنامه‌نویسی همروند وجود دارد، پیامدهای منفی حاصل از برنامه‌نویسی همروند، اتکاپذیری سیستم‌های همروند را به چالش می‌کشد. از طرفی رفع این پیامدها با روش‌های معمول وارسی و تست، در اغلب موارد منجر به کسب نتیجه بهینه و کارا نمی‌گردد. این امر به این خاطر است که عدم قطعیت و تعداد حالت‌های ممکن، در این‌گونه برنامه‌ها نسبت به برنامه های متوالی بسیار بیشتر است و با افزایش تعداد واحدهای همروند این عدم قطعیت به صورت نمایی افزایش پیدا می‌کند.

در حال حاضر عمده‌ترین چالشی که با رویکرد برنامه‌نویسی همروند مطرح شده، مسئله‌ی بروز بن‌بست، گرسنگی و شرایط رقابتی است، که به این گروه از مشکلات "خطاهای همروندی" اطلاق می‌شود. امروزه به منظور رفع خطاهای همروندی، با توجه به عدم کفایت راهکارهای رفع خطا و وارسی سنتی، نیازمند راهکارهای تازه و مؤثری هستیم. در این پایان‌نامه از مجموع خطاهای همروندی، به مسئله بن‌بست پرداخته شده است و راهکاری برای کشف بن‌بست‌های بالقوه در زمان اجرا، برای برنامه‌های همروند چندنخی ارائه گردیده است. این راهکار مبتنی بر استفاده از قابلیت شبکه‌های عصبی مصنوعی، در پیش‌بینی سری‌های زمانی است.

در راهکار پیشنهادی، نخست با نگاشت رفتارهای زمان اجرای واحدهای همروند به سری‌های زمانی، ساختار اطلاعاتی را ایجاد کرده‌ایم، که قابل پردازش باشد و در ادامه این ساختار اطلاعاتی را در شبکه‌های عصبی، که مبتنی بر نیازمندی‌های دامنه پیکربندی می‌شوند، برای پیش‌بینی بن‌بست مورد استفاده قرار دادیم. به منظور ارزیابی دقت راهکار پیشنهادی در پیش‌بینی بن‌بست، این راهکار را بر روی یک برنامه‌ی چندنخی نوشته شده به زبان جاوا ارزیابی کردیم. نتایج حاصل از این ارزیابی بیان‌گر کارایی راهکار ما با دقتی معادل ۷۳.۲۹ درصد بود که این دقت با در نظر داشتن ساختار تصادفی برنامه‌ی چندنخی مورد آزمون، عملکرد قابل توجهی می‌باشد.

کلمات کلی‌دی:

برنامه‌های همروند چندنخی، خطاهای همروندی، بن‌بست، شبکه عصبی مصنوعی، سری زمانی، پیش‌بینی سری زمانی

فهرست مطالب

فصل اول مقدمه و کارهای پیشین..... ۱

۱-۱.پیشگفتار..... ۲

۱-۲. اهداف پروژه..... ٤

۱-۳. ساختار پایان نامه..... ٤

۱-٤. بن‌بست و هزینه آن در سیستم‌های همروند..... ٥

۱-٤-۱. بن‌بست در سیستم‌های همروند..... ٥

۱-٤-۲.هزینه‌ی بن‌بست..... ۹

۱-٤-۲-۱. بن‌بست و سیستم‌های بحرانی..... ۹

۱-٤-۲-۱-۱. سیستم‌های بحرانی و نیازمندی‌های آن..... ۹

۱-٤-۲-۱-۲. هزینه بن‌بست در سیستم‌های بحرانی..... ۱۲

۱-٤-۲-۲. هزینه بن‌بست در سیستم‌های غیر بحرانی..... ۱٤

۱-٥. راهکارهای مقابله با بن‌بست..... ۱۷

۱-٥-۱. روش‌های سنتی برای حل مسئله بن‌بست..... ۱۹

۱-٥-۲. روش‌های ایستا برای کشف بن‌بست بالقوه..... ۲۱

۱-٥-۲-۱. روش‌های برمبنای علامت گذاری برای کشف بن‌بست‌های بالقوه و ابزارهای موجود..... ۲۱

۱-٥-۲-۱-۱. بررسی ابزار PDRFJ..... ۲۲

۱-٥-۲-۲. روش وارسی مدل برای کشف بن‌بست‌های بالقوه و ابزارهای موجود..... ۲٥

۱-٥-۲-۲-۱. ابزار Java PathFinder..... ۲٥

۱-٥-۳. روش‌های پویا برای حل مسئله بن‌بست..... ۲۸

۱-٥-۳-۱. راستی آزمایی زمان اجرا..... ۲۸

۱-٥-۳-۱-۱. مزایا و معایب راستی آزمایی در زمان اجرا..... ۲۹

۱-٥-۳-۱-۲. چگونگی پیاده سازی ناظر در برنامه هدف..... ۳۱

۱-٥-۳-۲. کشف بن‌بست‌های بالقوه در زمان اجرا..... ۳۳

۱-٥-۳-۲-۱. الگوریتم GoodLock..... ۳٤

۱-۵-۳-۲-۱-۱. مزایا و معایب الگوریتم GoodLock ۳۷
۱-۵-۳-۲-۲. گسترش یافته الگوریتم GoodLock ۳۸
۱-۵-۳-۲-۲-۱. مزایا و معایب گسترش یافته الگوریتم GoodLock ٤۰
۱-۵-۳-۲-۳. تحلیل برنامه به صورت پویا و تصادفی ٤۰
۱-۵-۳-۲-۳-۱. مزایا و معایب تحلیل برنامه به صورت پویا و تصادفی ٤۲
۱-۵-۳-۲-٤. آشکار سازی بن‌بست‌ها با تزریق نویز ٤۲
۱-۵-۳-۲-٤-۱. مزایا و معایب آشکار سازی بن‌بست‌ها با تزریق نویز ٤۳
۱-۵-۳-۲-۵. ایمنی از بن‌بست ٤۳
۱-۵-۳-۲-۵-۱. مزایا و معایب ایمنی از بن‌بست ٤۵
۱-۵-٤. مقایسه بین راهکارهای حل مشکل بن‌بست ٤۵
فصل دوم مدل پیشنهادی برای کشف بن‌بستهای بالقوه ٤۹
۲-۱. مقدمه ۵۰
۲-۲. مقدمه‌ای بر مدل پیشنهادی ۵۱
۲-۲-۱. گراف وابستگی ۵۳
۲-۲-۲. رفتارهای بن‌بست ساز ۵٤
۲-۳. معماری مدل پیشنهادی ۵٤
۲-۳-۱. مؤلفه اول : "Behavior extractor & Time series generator" ۵۵
۲-۳-۱-۱. بخش خارج از خط مؤلفه اول ۵٦
۲-۳-۱-۲. بخش برخط مؤلفه اول ۵۷
۲-۳-۱-۲-۱. وظایف تابع extractor&convertor() ۵۸
۲-۳-۱-۲-۱-۱. سری‌های زمانی ۵۹
۲-۳-۱-۲-۲. توضیح عملکرد کد تابع extractor&convertor() ٦۱
۲-۳-۲. مؤلفه دوم : "online lock tracker" ٦٤
۲-۳-۳. مؤلفه سوم : "predictor" ٦٦
۲-۳-٤. مؤلفه چهارم : "decision maker" ٦۸
۲-٤. نتیجه‌گیری ۷۱

فصل سوم بهبود مدل پیشنهادی با توجه به نیازمندی‌های مسئله‌ی بن‌بست..... ۷۳

۳-۱. مقدمه ۷٤

۳-۲. پیش‌بینی سری‌های زمانی ۷۵

۳-۲-۱. راهکار بر مبنای تابع مولد برای پیش‌بینی سری‌های زمانی..... ۷٦

۳-۲-۱-۱. روش Autoregressive(AR)..... ۷۷

۳-۲-۱-۲. روش Moving Average(MA)..... ۷۸

۳-۲-۱-۳. ترکیب روش MA و AR (ARMA)..... ۷۹

۳-۲-۱-٤. راهکار مدل مخفی مارکوف ۸۰

۳-۲-۲. راهکار هوش مصنوعی برای پیش‌بینی سری‌های زمانی ۸۰

۳-۲-۲-۱. شبکه‌های عصبی ۸۱

۳-۲-۲-۲. یادگیری در شبکه‌های عصبی..... ۸۳

۳-۲-۲-۲-۱. یادگیری نظارت شده ۸۳

۳-۲-۲-۲-۲. یادگیری نظارت نشده ۸۵

۳-۲-۲-۳. پیش‌بینی توسط شبکه‌های عصبی..... ۸٦

۳-۲-۳. ویژگی‌های مربوط به سری زمانی تولید شده در مدل پیشنهادی..... ۸۷

۳-۲-٤. مقایسه بین راهکارهای بر مبنای تابع مولد و هوش مصنوعی(انتخاب راهکار مناسب)..... ۸۹

۳-۲-٤-۱. شبکه‌های عصبی پیش‌بینی کننده..... ۹۱

۳-۲-٤-۱-۱شبکه‌های بازگشتی ۹۲

۳-۲-٤-۱-۲. شبکه‌های تاخیر زمانی ۹٤

۳-۲-٤-۱-۳. انتخاب نوع شبکه‌ی عصبی برای پیش‌بینی..... ۹۵

۳-۲-٤-۱-۳-۱. شبکه‌های عصبی Non-Linear Autoregressive..... ۹۵

۳-۲-٤-۱-۳-۲. شبکه عصبی Non-Linear Autoregressive with External input..... ۹٦

۳-۲-٤-۱-۳-۳. یادگیری در شبکه‌های عصبی NARX و NAR..... ۹۸

۳-۲-٤-۱-۳-٤. شبکه‌های عصبی NARX و سریهای زمانی آشوبی..... ۱۰۰

۳-۲-٤-۱-۳-۵. مرتبه پیچیدگی زمانی در شبکه‌های NAR و NARX..... ۱۰۲

۳-۳. ساختار دقیق مؤلفه پیش‌بینی کننده با توجه به شبکه‌ی عصبی انتخاب شده..... ۱۰۳

۳-۳-۱. تنظیم پارامتر میزان حافظه‌ی لازم برای شبکه‌ی NARX و NAR ۱۰۷
۳-۳-۲. مقایسه دقت پیش‌بینی با توجه به گسترش‌های پیشنهادی(NAR و NARX) ۱۰۸
۳-٤. آماده سازی اولیه مدل پیشنهادی ۱۱۹
۳-۵. پیاده سازی‌ها ۱۲۰
۳-٦. پیش‌بینی بن‌بست در زمان اجرا توسط مدل پیشنهادی ۱۲۱
۳-٦-۱. برنامه چندنخی مورد تست ۱۲۱
۳-٦-۲. نتایج آزمایش ۱۲۲
۳-۷. نتیجه‌گیری ۱۲۳
مراجع ۱۲۷
واژگان ۱۳۲
پیوست ها ۱٤٤

فهرست شکل‌ها

شکل ۱-۱یکی از اجراهای احتمالی دستور افزودن یک مقدار مشترک توسط نخ‌ها..... ۷

شکل ۱-۲ رابطه‌ی هزینه و اتکاپذیری..... ۱۱

شکل ۱-۳ مثال نخ‌ها در برنامه‌ی همروند..... ۱۸

شکل ۱-٤ معماری ابزار Java Path Finder..... ۲٦

شکل ۱-۵ اجرا و درخت قفل یک نخ..... ۳۵

شکل ۱-٦ مثالی از یک قفل درگاه..... ۳٦

شکل ۱-۷ شبه کد الگوریتم GoodLock..... ۳٦

شکل ۱-۸ مثال برای الگوریتم GoodLock..... ۳۷

شکل ۱-۹ تبدیل درخت قفل به گراف جهت دار..... ۳۹

شکل ۱-۱۰ معماری ایمنی از بن‌بست..... ٤٤

شکل ۲-۱ مثالی از اجرای دو نخ..... ۵۲

شکل ۲-۲ مثال گراف وابستگی..... ۵٤

شکل ۲-۳ معماری مدل پیشنهادی..... ۵۵

شکل ۲-٤ شبه کد تابع extractor&convertor..... ۵۷

شکل ۲-۵ بدنه‌ی تابع ()extractor&convertor..... ٦۲

شکل ۲-٦ بدنه‌ی تابع ()thisPeriodBehavior..... ٦۳

شکل ۲-۷ تابع ()OverallBehaviors..... ٦۳

شکل ۲-۸ گراف وابستگی برای مثال..... ٦٤

شکل ۲-۹ گراف وابستگی به روز شده برای مثال..... ٦۵

شکل ۲-۱۰ گراف وابستگی به روز شده برای مثال..... ٦۵

شکل ۲-۱۱ گراف وابستگی به روز شده برای مثال..... ٦٦

شکل ۲-۱۲ پیش‌بینی انجام شده برای مثال..... ٦۷

شکل ۲-۱۳ گراف وابستگی به روز رسانی شده توسط مؤلفه "online lock tracker" در انتهای پریود چهارم..... ٦۹

شکل ۲-۱٤ گراف وابستگی انتزاعی تولید شده ۷۰
شکل ۲-۱۵ گراف وابستگی انتزاعی تولید شده ۷۱
شکل ۳-۱ شبکه‌ی عصبی جانداران(سمت چپ)، شبکه‌ی عصبی مصنوعی(سمت راست) ۸۱
شکل ۳-۲ یک شبکه‌ی عصبی نمونه ۸۲
شکل ۳-۳ شبه کد الگوریتم back-propagation ۸٤
شکل ۳-٤ پیش‌بینی سری‌های زمانی توسط شبکه‌ی عصبی ۸٦
شکل ۳-۵ تهیه مجموعه‌ی آموزشی از روی سری زمانی ۸۷
شکل ۳-٦ استفاده از سری زمانی خارجی در پیش‌بینی یک سری زمانی ۹۱
شکل ۳-۷ شبکه‌ی المن ۹۳
شکل ۳-۸، ساختار عمومی شبکه‌های عصبی بازگشتی ۹٤
شکل ۳-۹ شبکه‌ی پویای تاخیر زمانی ۹٤
شکل ۳-۱۰ شبکه‌ی NAR ۹۷
شکل ۳-۱۱ شبکه‌ی NARX[44] ۹۷
شکل ۳-۱۲ ساختار دقیق‌تر شبکه‌ی NARX ۹۸
شکل ۳-۱۳ شبکه‌ی بریده شده‌ی NARX جهت فاز آموزش ۹۹
شکل ۳-۱٤ سری زمانی Mackey-Glass[47] ۱۰۰
شکل ۳-۱۵ سری زمانی weirstrass[47] ۱۰۱
شکل ۳-۱٦ سری زمانی BET[47] ۱۰۱
شکل ۳-۱۷ معماری مدل پیشنهادی ۱۰۳
شکل ۳-۱۸ خروجی مؤلفه اول ۱۰۵
شکل ۳-۱۹ ساختار داخلی مؤلفه پیش‌بینی کننده با استفاده از شبکه NARX با توجه به مثال ۱۰۵
شکل ۳-۲۰ ساختار درونی مولفه پیش‌بینی کننده در صورتی که از شبکه NAR استفاده شود ۱۰۵
نمودار ۳-۱. نمودار مربوط به نرخ شکست براساس جداول ۳-۱۱ تا ۳-۱٤ (محور افقی "نام جدول" و محور عمودی "نرخ شکست") ۱۱۵
نمودار۳-۲. مربوط به نرخ شکست براساس جداول ۳-۱۵ تا ۳-۱۸ (محور افقی "نام جدول" و محور عمودی "نرخ شکست") ۱۱۷

نمودار ۳-۳. نشان دهنده میزان متوسط خطا برای ۱۰۰ تا ۱۰۰۰ بار اجرا با مقدار داده‌های اعتبار سنجی ۱۵٪..... ۱۱۸
نمودار۳-٤. نشان دهنده میزان متوسط خطا برای ۱۰۰ تا ۱۰۰۰ بار اجرا با مقدار داده‌های اعتبار سنجی ۲۵٪..... ۱۱۹

فهرست جداول

جدول ۳-۱ نتایج پیش‌بینی زنجیره مارکوف با تقسیم‌بندی (۱۵و۲۵و۶۰) ۱۱۰
جدول ۳-۲ نتایج پیش‌بینی زنجیره مارکوف با تقسیم‌بندی (۲۰و۲۵و۵۵) ۱۱۱
جدول ۳-۳ نتایج پیش‌بینی زنجیره مارکوف با تقسیم‌بندی (۳۰و۲۵و۴۵) ۱۱۱
جدول ۳-٤ نتایج پیش‌بینی زنجیره مارکوف با تقسیم‌بندی (٤۰و۲۵و۳۵) ۱۱۱
جدول ۳-۵ نتایج پیش‌بینی زنجیره مارکوف با تقسیم‌بندی (۵۰و۲۵و۲۵) ۱۱۱
جدول ۳-٦ نتایج پیش‌بینی زنجیره مارکوف با تقسیم‌بندی (۱۵و۱۵و۷۰) ۱۱۱
جدول ۳-۷ نتایج پیش‌بینی زنجیره مارکوف با تقسیم‌بندی (۲۰و۱۵و٦۵) ۱۱۲
جدول ۳-۸ نتایج پیش‌بینی زنجیره مارکوف با تقسیم‌بندی (۳۰و۱۵و۵۵) ۱۱۲
جدول ۳-۹ نتایج پیش‌بینی زنجیره مارکوف با تقسیم‌بندی (٤۰و۱۵و٤۵) ۱۱۲
جدول ۳-۱۰ نتایج پیش‌بینی زنجیره مارکوف با تقسیم‌بندی (۵۰و۱۵و۳۵) ۱۱۲
جدول ۳-۱۱ نتایج پیش‌بینی توسط شبکه‌ی عصبی با تقسیم‌بندی (۱۵و۲۵و٦۰) ۱۱۳
جدول ۳-۱۲ نتایج پیش‌بینی توسط شبکه‌ی عصبی با تقسیم‌بندی (۲۰و۲۵و۵۵) ۱۱۳
جدول ۳-۱۳ نتایج پیش‌بینی توسط شبکه‌ی عصبی با تقسیم‌بندی (۳۰و۲۵و٤۵) ۱۱۳
جدول ۳-۱٤ نتایج پیش‌بینی توسط شبکه‌ی عصبی با تقسیم‌بندی (٤۰و۲۵و۳۵) ۱۱٤
جدول ۳-۱۵ نتایج پیش‌بینی توسط شبکه‌ی عصبی با تقسیم‌بندی (۱۵و۱۵و۷۰) ۱۱۵
جدول ۳-۱٦ نتایج پیش‌بینی توسط شبکه‌ی عصبی با تقسیم‌بندی (۲۰و۱۵و٦۵) ۱۱۵
جدول ۳-۱۷ نتایج پیش‌بینی توسط شبکه‌ی عصبی با تقسیم‌بندی (۳۰و۱۵و۵۵) ۱۱٦
جدول ۳-۱۸ نتایج پیش‌بینی توسط شبکه‌ی عصبی با تقسیم‌بندی (٤۰و۱۵و٤۵) ۱۱٦
جدول ۳-۱۹ نتایج آزمایش پیش‌بینی بن‌بست در برنامه‌ی همروند هدف ۱۲۲

فصل اول

مقدمه و کارهای پیشین

۱. مقدمه و کارهای پی‌شی‌ن

۱-۱. پی‌شگفتار

افزایش نیاز به سرعت محاسباتی بالا در کاربردهای کامپیوتری، طراحان سیستم‌های کامپیوتری را به سوی تولید پردازنده‌های چند هسته‌ای[1] هدایت کرد. این اندیشه زمانی به مرحله اجرا درآمد که به دلیل مشکلات متعددی از جمله بالا بودن درجه حرارت پردازنده، امکان افزایش فرکانس در ساعت[2] پردازنده وجود نداشت. بنابراین برای افزایش سرعت در سیستم‌های کامپیوتری، رویکرد جدیدی با عنوان پردازنده‌های چند هسته‌ای مطرح شد که هدف از آن تقسیم کار بین هسته‌ها به جای استفاده از یک هسته پرسرعت بود. اولین پردازنده-ی دو هسته‌ای در سال ۲۰۰۵ میلادی توسط شرکت ADM با نام Athlon 64 وارد بازار شد[۱].

بعد از آن، شرکت‌های Intel و IBM به صف تولیدکنندگان پردازنده‌های چند هسته‌ای پیوستند. در تئوری، اضافه کردن هر یک هسته، باید افزایش سرعتی معادل با دوبرابر اولیه داشته باشد، اما در عمل این اتفاق نمی افتد. در واقع با ورود پردازنده‌های چندهسته‌ای مشکلات تازه‌ای مطرح شد که قبلا وجود نداشتند. این مشکلات هم در حوزه سخت افزار و هم در حوزه نرم افزار سکوی جدیدی برای تحقیقات گشود. در حوزه سخت افزار مشکلاتی از قبیل:

- چگونه هسته‌ها با یکدیگر ارتباط داشته باشند؟
- آیا همه پردازنده‌ها باید یکسان باشند؟

1 Multicore Processor
2 Clock

- تعریف دستورات پایه باید به همان شکل قبل باشد یا لازم است تا تغییر کند؟

معماران کامپیوتر را به سوی راهکارهایی مانند ساختار خط لوله ای[1]، فرستادن از پیش[2] و غیره، هدایت کرد که نتیجه آن بالا رفتن سرعت پردازنده‌های چندهسته‌ای علی رغم مشکلات بود[۱]. اما در حوزه نرم افزار، مهمترین سوالی که مطرح شد این بود که آیا برنامه نویسان قادر خواهند بود برنامه‌هایی بنویسند که از قابلیت چندهسته‌ای استفاده کند یا خیر؟[۱] برنامه نویسی چندنخی[3] اولین پاسخ برای این سوال بود. برنامه نویسی چندنخی مفهومی بود که قدمت آن به قبل از تولید پردازنده‌های چندهسته‌ای بازمیگردد. هدف اصلی از ایجاد برنامه‌های چندنخی، تقسیم یک مسیر اجرایی بین چندین نخ اجرایی بود. نخ یک واحد اجرایی است که به تنهایی قابلیت شروع، اجرا و خاتمه را دارد. در استفاده‌های اولیه از برنامه نویسی چندنخی، از این روش برای کاهش زمان پاسخ دهی استفاده میشد. به این صورت که به هر نخ یک زمان اجرا اختصاص داده می‌شود و پردازنده با توجه به یک الگوریتم خاص و زمان اختصاص داده شده به هر نخ، این نخ‌ها را به شکل نوبتی اجرا میکرد. مزیت این روش این بود که تمام نخ‌ها فرصت اجرا پیدا میکردند و در هر بار اجرا بخشی از یک نخ اجرا میشد، در نتیجه همه کارها تقریبا با هم پیش میرفت. با ورود پردازنده‌های چندهسته ای، استفاده از برنامه نویسی چندنخی موضوعیت بیشتری پیدا کرد. در این حالت امکان این وجود داشت که نخ‌ها بر روی هسته‌های مختلف به صورت همزمان باهم اجرا شوند. در این شرایط دیگر استفاده از برنامه نویسی چند نخی فقط راهکاری برای بهینه کردن زمان پاسخ دهی نبود بلکه ابزاری بود که برای کاربرد داشتن و قابل استفاده بودن پردازنده‌های چندهسته‌ای باید مورد استفاده قرار میگرفت. اهمیت و نیاز به استفاده از برنامه نویسی چندنخی، محققان را بر روی مشکلاتی که با برنامه نویسی چندنخی وارد حوزه نرم افزار شده بود، متمرکز ساخت. از اولین مشکلاتی که برنامه نویسان سیستم‌های همروند با آن مواجه شدند عبارت بودند از: شرایط رقابتی، گرسنگی و بن‌بست. برای حل هرکدام از این مشکلات روش‌ها و راهکارهای برخط[4] یا غیربرخط[1] پیشنهاد شد که هر کدام در حوزه مسائل خاصی کاربرد داشته‌اند.

1 Pipeline
2 Feed-Forward
3 Multithread Programming
4 Online

۱-۲.اهداف پروژه

همان‌طور که اشاره شد، بن‌بست یکی از مشکلاتی است که با ورود برنامه نویسی همروند در حوزه نرم افزار و برنامه نویسی مطرح گردید. راهکارهای بسیاری برای حل این مسئله پیشنهاد شده‌اند که به طور کلی به راهکارهای: زمان برنامه نویسی، زمان کامپایل یا تفسیر، زمان اجرا و یا ترکیبی از آنها، تقسیم‌بندی میکنیم. با در نظر گرفتن روش‌های ارائه شده در هرکدام از این دسته‌ها، اهداف اصلی که در این پروژه دنبال می‌شود، به طور خلاصه در ذیل عنوان شده است:

۱. ارزیابی و مقایسه: ارزیابی روش‌های مختلفی که به نوعی سعی در حل مشکل بن‌بست دارند و بررسی کاربرد آنها در حوزه‌های مختلف و مقایسه نقاط ضعف و قوت آنها با یکدیگر.
۲. ارائه مدل پشنهادی: ارائه مدلی جدیدی برای رفع مسئله بن‌بست در برنامه‌های همروند چندنخی با این هدف که نقاط ضعف روش‌های موجود را پوشش دهد.
۳. بهبود عملکرد مدل پیشنهادی: با توجه به گسترش پذیری مدل پیشنهادی، نشان داده خواهد شد که قابلیت بهبود عملکرد مدل، با توجه به شرایط مسئله همواره وجود دارد.

۱-۳.ساختار پایان نامه

در ادامه‌ی این فصل ابتدا به بررسی ویژگی‌های سیستم‌های همروند چندنخی، کاربرد و حساسیت آنها می‌پردازیم و با مشکلاتی که بن‌بست ممکن است برای یک سیستم همروند به وجود آورد با جزئیات بیشتری آشنا می‌شویم. سپس روش‌های مختلفی که برای حل مشکل بن‌بست پیشنهاد شده است را عنوان کرده و تحقیقات و کارهای عملی که در هر مورد انجام شده است را با نگاهی جزئی نگرانه بررسی و نتیجه‌گیری خواهیم کرد.

1 Offline

در فصل دوم روش پیشنهادی برای کشف بن‌بست‌های بالقوه در برنامه‌های همروند چندنخی با کمک راهکارهای پیش‌بینی مطرح خواهد شد و ویژگی‌های مثبت و منفی آن با روش‌های موجود برای حل مسئله بن‌بست مقایسه خواهد شد.

در فصل سوم به بررسی گزینه‌های موجود در حوزه پیش‌بینی رفتار برنامه‌های همروند پرداخته و از بین آنها مناسب‌ترین مورد را برای به کارگیری در مدل پیشنهادی خود انتخاب خواهیم کرد. در این فصل آزمایش‌هایی نیز برای مقایسه نتایج مدل پیشنهادی با سایر روش‌های موجود نیز آورده شده است.

در انتهای پایان نامه چند ضمیمه نیز اضافه شده است. ضمائم شامل جزئیات پیاده سازی و واژه‌نامه می‌باشند.

۱-۴.بن‌بست و هزی‌نه آن در سی‌ستم‌های همروند

همان‌طور که اشاره شد، بن‌بست یکی از عمده مشکلات سیستم‌های همروند است. این مشکل چه در سیستم‌های حافظه مشترک مثل برنامه‌های چندنخی و چه در سیستم‌های توزیع شده مثل برنامه‌های انتقال پیام، رخ می‌دهد. در ادامه مروری خواهیم داشت بر مشکلات و چگونگی رخداد بن‌بست و نیز هزینه‌هایی که وقوع بن‌بست به سیستم‌های همروند تحمیل می‌کند.

۱-۴-۱. بن‌بست در سی‌ستم‌های همروند

در بیشتر کاربردهایی که از برنامه نویسی چندنخی وجود دارد، هدف استفاده از چند واحد اجرایی است که این واحدها با یکدیگر برای انجام هدف مشترکی همکاری می‌کنند. بنابراین اولین و مهمترین موضوعی که مطرح می‌شود اینست که نخ‌ها چگونه با یکدیگر ارتباط برقرار کنند. مرسوم ترین روش در برنامه نویسی چندنخی برای ارتباط بین نخ‌ها استفاده از برنامه نویسی چند نخی با حافظه مشترک است. در این مدل تمامی نخ‌ها به فضای حافظه مشترکی دسترسی دارند. مزیت این روش در اینست که کار برنامه نویسی چندنخی آسان تر شده و تقریبا مشابه برنامه نویسی ترتیبی می‌شود. اما این مزیت آغاز عمده مشکلات برنامه

نویسی چندنخی حافظه مشترک نیز می‌باشد[۲]. از آنجایی که همه نخ‌ها به همه حافظه مشترک دسترسی دارند، بنابراین در استفاده از حافظه مشترک باید دقت بیشتری نسبت به برنامه نویسی ترتیبی لحاظ کرد. عمده مشکل برنامه نویسی چندنخی حافظه مشترک مسئله رقابت می‌باشد. به وجود آمدن شرایط رقابتی[1] در برنامه چندنخی به احتمال زیادی منجر به رفتار نادرست برنامه در نتیجه منجر به خروجی نادرست خواهد شد. برای واضح شدن موضوع از یک مثال سنتی استفاده میکنیم.

فرض کنید برنامه چند نخی نوشته شده است که وظیفه آن دریافت درخواست‌ها و پردازش درخواست و به روز رسانی مجموع تعداد درخواست‌ها می‌باشد. همچنین فرض کنید که هر نخ مسئول دریافت یک درخواست، پردازش آن و اضافه کردن یک متغییر عمومی به نام totalRequests توسط دستور زیر است:

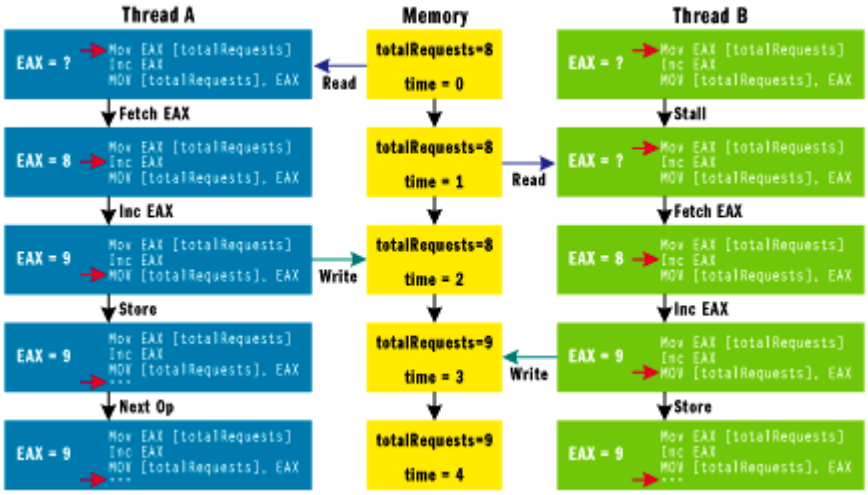
$$totalRequests = totalRequests + 1$$

کامپایلر دستور فوق را تبدیل به کد ماشین زیر خواهد کرد:

```
MOV EAX, [totalRequests]   // load memory for totalRequests into
register
INC EAX                    // update register
MOV [totalRequests], EAX   // store updated value back to memory
```

اگر دو نخ به طور همزمان این کد را اجرا کنند، ممکن است حالتی مشابه آنچه که شرحش در ادامه آمده است، پیش بیاید. همان طور که از شکل ۱-۱ پیداست، هر دو نخ به طور همزمان یک مقدار از totalrequst را میخوانند، هر دو یک واحد آن را افزایش می‌دهند و آن را دوباره ذخیره می‌کنند. به این ترتیب دو درخواست توسط نخ‌ها انجام شده است اما فقط یک واحد به مجموع کل افزوده شده است.

1 Race Condition



شکل ۱-۱یکی از اجراهای احتمالی دستور افزودن یک مقدار مشترک توسط نخ‌ها

به خطاهایی از این دست که به خاطر زمان بندی اجرای نخ‌ها اتفاق می‌افتد، شرایط رقابتی گفته می‌شود. برای به وجود آمدن شرایط رقابتی در برنامه‌های چند نخی چهار شرط لازم است:

۱. باید مکان‌هایی از حافظه وجود داشته باشد که توسط دو یا تعداد بیشتری نخ قابل دسترسی باشند.
۲. باید این مکان مشترک حافظه ویژگی داشته باشد که عملکرد درست برنامه منوط به صحیح بودن مقدار ان ناحیه از حافظه باشد.
۳. در طول مدت زمانی که این ناحیه از حافظه در حال برروز رسانی می‌باشد، اختصاصی نشده باشد.
٤. درست در زمانی که شرط سوم برقرار است، نخ دیگری بتواند به آن ناحیه از حافظه دسترسی داشته باشد.

اگر تمامی این شرایط برقرار باشد به احتمال زیاد درستی برنامه موازی دچار مشکل خواهد شد. برای حل مشکل شرایط رقابتی از جمله موثرترین و پراستفاده ترین روش‌ها استفاده از راهکارهای قفل‌گذاری[1] است. قفل‌ها از دسترسی بقیه نخ‌ها به فضای حافظه‌ای که نخ دیگری در حال به روز رسانی آن می‌باشد، جلوگیری می‌کنند. این عمل شرط چهارم از

1 Lockning Mechanism