برنامه‌نویسی وظیفه‌ای در
.NET 4.0 & 4.5

با استفاده از

TPL & PLINQ

مهندس سید منصور عمرانی

انتشارات پندار پارس

سرشناسه : عمرانی، سید منصور، ۱۳۵٦ -
عنوان و نام پدیدآور : برنامه‌نویسی وظیفه‌ای در NET 4.0. و NET 4.5. با استفاده از TPL و PLINQ : چگونه برنامه‌هایی سریع‌تر و کاربرپسندتر بنویسیم/ ترجمه و تالیف سید منصور عمرانی.
مشخصات نشر : تهران : پندار پارس ، ۱۳۹۱ .
مشخصات ظاهری : ٦۹٦ ص.: مصور، جدول.
شابک : 978-600-6529-12-7: ۱۹۸۰۰۰ریال
موضوع : مایکروسافت ویژوال سی شارپ دات نت
موضوع : سی شارپ (زبان برنامه‌نویسی کامپیوتر)
موضوع : برنامه‌نویسی شی‌گرا
موضوع : برنامه‌نویسی موازی
رده بندی کنگره : QA٧٦/٧٣ /س۱۳۹۱۹۵ ۸۳ع
رده بندی دیویی : ۱۳۳/۰۰۵
شماره کتابشناسی ملی : ۲۷۳۵۰۱۷

انتشارات پندارپارس

دفتر فروش: انقلاب، ابتدای کارگرجنوبی، کوی رشتچی، شماره ١٤، واحد ١٦ **www.pendarepars.com**
تلفن: ٦٦٥٧٢٣٣٥ - تلفکس: ٦٦٩٢٦٥٧٨همراه: ۰۹۱۲۲٤٥۲۳٤۸ **info@pendarepars.com**

نام کتاب : برنامه‌نویسی وظیفه‌ای در NET 4.0 & 4.5. با استفاده از TPL & PLINQ
ناشر : انتشارات پندار پارس ناشر همکار: پارشمن
ترجمه و تالیف : سید منصور عمرانی
چاپ نخست : بهار ۹۱
شمارگان : ۱۰۰۰ نسخه
طرح جلد : رامین شکرالهی
لیتوگرافی، چاپ، صحافی : ترام‌سنج، صالحان، خیام
قیمت : ۱۹۸۰۰ تومان به همراه CD شابک : ۷-۱۲-٦٥۲۹-٦۰۰-۹۷۸

مقدمه

دنیای برنامه‌نویسی و نرم‌افزار همیشه دستخوش تغییر و تحول‌های جدید بوده است. از معرفی کلاس‌ها و مولفه‌های جدید گرفته تا معرفی فناوری‌های جدید نرم‌افزاری و شیوه‌ها و الگوهای جدید برنامه‌نویسی. در این میان یکی از تحولات تاثیرگذار بر دنیای برنامه‌نویسی در سال‌های اخیر تغییر و تحول پردازنده‌ها و سکوهای سخت‌افزاری بوده است. به طوری که دیگر عصر تولید پردازنده‌های تک هسته‌ای به سر رسیده و عصر پردازنده‌های چند هسته‌ای و چندپردازنده‌ها آغاز شده است.

این تغییر به نوبه‌ی خود شرکت‌های رهبری کننده‌ی فناوری‌های نرم‌افزار را به تکاپو انداخته است. زیرا اگر کسی علاقه دارد برنامه‌اش در سکوهای جدید سریع‌تر کار کند، باید دیدگاه برنامه‌نویسی خود را تغییر بدهد و موازی فکر کند. علاوه بر این پاسخ‌پذیر بودن برنامه‌ها در ادواتی مانند Tablet PC یا Touch بسیار مهم است و راه حل آن نیز فقط اجرای موازی است. در واقع تا جای ممکن باید کارهای زمان‌بر را که باعث قفل شدن موقت واسط کاربر یا یخ زدن فرم‌ها می‌شود به صورت موازی اجرا کنیم.

در این میان شرکت مایکروسافت با معرفی کتابخانه‌ی اختصاصی برنامه‌نویسی موازی به نام TPL در NET 4.0. گوی سبقت را از دیگر شرکت‌ها ربوده است. خصوصاً اضافه شدن کلمات کلیدی async و await به زبان‌های NET. و مجموعه‌ی وسیعی از متدهای غیر همزمان به کتابخانه‌ی کلاس‌ها در NET 4.5. نشان می‌دهد مایکروسافت توجه و عظمی جدی به این مقوله دارد. با این اوصاف می‌توان گفت آشنایی با TPL به نوعی اجتناب‌ناپذیر است و کسانی که می‌خواهند در NET 4.5. برنامه‌نویسی کنند حتماً باید با TPL و الگوی TAP آشنا باشند.

محتوای کتاب

کتاب «برنامه‌نویسی وظیفه‌ای در NET 4.0 & 4.5. با استفاده از TPL و PLINQ» حاصل 8 ماه تلاش و زحمت مستمر در خصوص شیوه‌ی جدید برنامه‌نویسی موازی در NET 4.0. و NET 4.5. است. محتوای کتاب ترکیبی از ترجمه، تالیف و تحقیق است. مولف در نوشتن کتاب از 12 کتاب لاتین استفاده نموده و با تحقیق و مطالعه‌ی مقالات بیشماری در اینترنت و مستندات MSDN، مجموعه‌ای را گردآوری کرده که بدون اغراق می‌توان گفت از نظر جامعیت مشابه ندارد. منابع اصلی مولف در تالیف این کتاب به شرح زیر بوده است:

#	عنوان	ناشر	مولف	سال انتشار
1	Essential C# 4.0	Addison-Wesley	Mark Michaelis	2010
2	Pro .NET 4.0 Parallel Programming in C#	Apress	Adam Freeman	2010
3	Accelerated C# 2010	Apress	Trey Nash	2010
4	Pro C# 2010 and .NET 4.0 Platform	Apress	Andrew Troelsen	2010
5	C# 4.0 The Complete Reference	McGraw-Hill	Herbert Schildt	2010
6	CLR via C# 3rd Edition	MicrosoftPress	Jeffrey Richter	2010
7	C# 4.0 in a Nutshell	OReilly	Josep Albahari Ben Albahari	2010
8	Practical .NET2 and C# 2	ParadoxalPress	Patrick Smacchia	2006
9	Professional Parallel Programming with C#	Wrox	Gaston C.Hillar	2011

#	عنوان	ناشر	مولف	سال انتشار
10	Windows Internals 5th Edition	MicrosoftPress	Mark Russinovich David Solomon	2009
11	Essential COM	Addison-Wesley	Don Box	1997
12	Patterns of Parallel Programming in .NET (C#)	Stepehn Toub	Microsoft	2010
13	Task-Based Asynchronous Pattern	Stephen Toub	Microsoft	2010

مخاطبین

مخاطبین این کتاب تمام کسانی هستند که به طور کلی تحت سکوی .NET برنامه‌نویسی می‌کنند، زیرا برنامه‌نویسی موازی منحصر به نوع به خصوصی از برنامه‌ها نیست. در این کتاب شیوه‌ی قدیمی و سنتی و همچنین شیوه‌ی مدرن برنامه‌نویسی موازی در .NET 4.0 و .NET 4.5 با استفاده از مثال‌های عملی متعدد به صورت اصولی و زیر بنایی آموزش داده می‌شود. این کتاب خصوصاً برای برنامه‌نویسان برنامه‌های تحت ویندوز و همچنین افرادی که قصد افزایش کارایی برنامه‌های تحت سرویس‌دهنده‌ی خود را دارند بسیار مفید است. علاوه بر این کتاب مزبور می‌تواند برای دانشجویان رشته‌ی مهندسی نرم‌افزار در زمینه‌ی درس سیستم عامل، برنامه‌نویسی موازی و تا حدی کامپایلر مفید باشد.

نیازمندی‌ها

خواننده برای مطالعه و فراگیری مباحث این کتاب به داشتن تجربه یا تخصصی ویژه و طولانی در کار با .NET نیازی ندارد. اما باید با اصول کلی برنامه‌نویسی و شیءگرایی آشنا باشد. زبان مقصد این کتاب #C است. لذا خواننده باید در حد متوسط با این زبان آشنا باشد و بتواند یک برنامه‌ی .NET را در Visual Studio اجرا کند. مثال‌های این کتاب عمدتاً برنامه‌های ساده‌ی Console Application هستند، اما گهگاه از برنامه‌های Windows Forms نیز استفاده شده است. با توجه به این مساله خواننده باید آشنایی نسبی و مختصری با نوشتن یک برنامه‌ی تحت ویندوز توسط Visual Studio را داشته باشد.

همچنین بحث این کتاب به نسخه‌ی خاصی از .NET منحصر نمی‌شود و تمامی نسخه‌های .NET را در بر می‌گیرد. اما سعی شده هر جا کلاس یا قابلیتی به نسخه‌ی به خصوصی از .NET نیاز داشته باشد، شماره‌ی نسخه‌ی .NET ذکر شود. در صورتی که خواننده از Visual Studio 2010 استفاده کند می‌تواند تمام مثال‌های کتاب را (به جز مثال‌های فصل 34 و 35) بدون هیچ مشکلی اجرا کند. مابقی کُدها (فصل 34 و 35) به قابلیت‌های خاص .NET 4.5 بر می‌گردد که با توجه به این که هنوز این نسخه از .NET به طور رسمی توسط مایکروسافت ارائه نشده نمی‌توان آنها را با استفاده از Visual Studio 2010 اجرا کرد. اما خواننده می‌تواند با دانلود نسخه‌ی آزمایشی Visual Studio 11 آنها را آزمایش کند و از قبل خود را مهیای قابلیت‌های برنامه‌نویسی موازی .NET 4.5 نماید.

ساختار کتاب

این کتاب در قالب 6 بخش و 40 فصل تالیف شده و هر فصل به مبحث مجزایی می‌پردازد. شیوه‌ی ارائه‌ی مطلب در این کتاب، آموزش عملی همراه با مثال است. لذا هر فصل با مثال‌های زیادی همراه است اما با توجه به تعدد موضوعات مطرح شده، این کتاب را به عنوان مرجع نیز می‌توان استفاده کرد. فصل‌های کتاب بر اساس ترتیب مشخصی از لحاظ یادگیری و آموزش چیده شده به طوری که هر فصل نوعاً پیش نیاز فصل بعدی محسوب می‌شود.

به همین دلیل روش مورد انتظار مطالعه‌ی کتاب، خواندن آن از ابتدا تا انتها بر اساس شماره‌ی فصل‌ها است. با این وجود اگر خواننده با برخی مباحث آشنا باشد و بخواهد یک فصل به خصوص را مطالعه کند می‌تواند مستقیماً به آن مراجعه کند و الزامی وجود ندارد حتماً فصل‌های قبل از آن را هم بخواند (اگرچه توصیه می‌شود این کار را بکند). اما در هر فصل فرض می‌شود خواننده قبلاً با مباحث پیش‌نیاز آن آشنایی داشته باشد

موضوع فصل‌ها نیز از یکدیگر کاملاً مجزا است و از ارائه‌ی چندین موضوع در یک فصل خودداری شده است. به همین دلیل است که تعداد فصول کتاب زیاد است. اما مزیت این شیوه‌ی تقسیم‌بندی این است که خواننده به سرعت می‌تواند به موضوع خاصی در کتاب مراجعه کرده و آن را مطالعه نماید. تلاش بسیاری صورت گرفته تا محتوای هر فصل کاملاً جامع باشد و تمام نکات مرتبط با موضوع را پوشش بدهد. به طوری که بعد از خواندن آن چیز دیگری باقی نماند که خواننده باید بداند. سعی شده فصل‌ها تا جای ممکن کوتاه باشد، با این حال در برخی از فصول نظیر فصول 29 تا 32 که به الگوهای برنامه‌نویسی غیر همزمان مربوط می‌شود، برای ادا شدن حق مطلب محتوا کمی مفصل است.

سی‌دی ضمیمه

این کتاب مشتمل بر حدود 200 مثال عملی است که تمام آنها در قالب پروژه‌های مجزا و قابل اجرا در سی‌دی ضمیمه قرار داده شده است. علاوه بر این می‌توانید مثال‌های کتاب را در صفحه‌ی اختصاصی آن از سایت انتشارات پندار پارس دانلود نمایید.

بازخورد

پرداختن به مقوله‌ی برنامه‌نویسی چندنخی و موازی کار بسیار سختی است و نوشتن یک کتاب جامع در این باره بسیار سخت‌تر است. مولف با خواندن کتاب‌های مختلف و مقالات و مستندات بیشمار در رابطه با هر موضوع، حداکثر تلاش خود را به کار برده تا به طور کاملاً جامع به تمامی مباحث برنامه‌نویسی موازی در سکوی .NET بپردازد. لذا از طرح ابهام‌آمیز یا ناقص مطلب خودداری نموده است. قطعاً این کتاب خالی از اشکال نیست ضمن این که موضوعات دیگری نیز وجود دارد که در این کتاب قابل طرح است. حتماً نظرات، انتقادات، سوالات و پیشنهادات خود را در رابطه با این کتاب با مولف از طریق سایت انتشارات (www.pendarepars.com) یا آدرس mansoor.omrani@yahoo.com در میان بگذارید. مولف در انتها امیدوار است این کتاب برای جامعه‌ی برنامه‌نویسان مفید بوده و از خواندن آن لذت ببرید.

سید منصور عمرانی
بهار 1391

تقدیم به پدر و مادری مهربان

که محبتی بی دریغ نثارم کردند

فهرست کتاب در یک نگاه

فهرست

بخش 1
برنامه‌نویسی چندنخی سنتی

فصل 1. تاریخچه‌ی پردازنده‌ها و برنامه‌نویسی هم‌روند

غول بی‌شاخ و دم برنامه‌نویسی چندنخی

وقت اسم برنامه‌نویسی چند نخی به میان می‌آید بسیاری از برنامه‌نویسان وحشت می‌کنند، زیرا نام این شیوه‌ی برنامه‌نویسی همیشه با یک جور اضطراب و استرس همراه است. مهم نیست چه احساسی نسبت به این مساله دارید، اما واقعیت این است که برنامه‌نویسی هم‌روند مثل راه رفتن در یک میدان پر از مین است! بدون این که علتش را بفهمید ممکن است برنامه‌ای که چندین بار آن را آزمایش کرده بودید و درست کار می‌کرد باگ پیدا کند یا جور دیگری کار کند. باگ‌های برنامه‌های هم‌روند هم یکی از بدخیم‌ترین و پیچیده‌ترین نوع باگ‌ها هستند و تشخیص یا رفع آنها کار بسیار مشکلی است.

اوضاع هم زمانی بدتر می‌شود که تعداد پردازنده‌ها و هسته‌ها بیشتر شود. زیرا برخی از باگ‌های بدذات و موذی تا زمانی که برنامه را روی یک سیستم چندهسته‌ای یا چندپردازنده‌ای اجرا نکنید خودشان را نشان نمی‌دهند. تری نَش در کتاب Accelarated C# 2010 داستانی را از تجربه‌ی خودش ذکر می‌کند که قرار بوده نسخه‌ی نهایی یک برنامه را برای شرکت توزیع کننده بفرستند تا آن را بر روی دیسک در تیراژ وسیع تکثیر کند و برای فروش به بازار بفرستد. درست در آخرین لحظات قبل از ارسال برنامه برای شرکت توزیع کننده یکی از برنامه‌نویسان برنامه را روی یک کامپیوتر چند پردازنده اجرا می‌کند و همان لحظه سر و کله‌ی یک باگ پیدا می‌شود.

او می‌گوید آن ماجرا درس بسیار بزرگی برای او و سایر برنامه‌نویسان تیم‌شان بوده است. با این اوصاف شما یا آدم با دل و جراتی هستید و یا خبر ندارید وارد چه سرزمین پر خطری شده‌اید. اما به شما بگویم اگر قدم در این جاده گذاشته‌اید سعی کنید تا آخرش بروید. نصفه و نیمه کار کردن و نوشتن برنامه‌های چند نخی با دانش کم بدترین کاری است که می‌توانید با خودتان بکنید.

آغاز سفر

فکر می‌کنم به عنوان یک پیش در آمد یا مقدمه واقعاً شما را ترسانده باشم. واقعیتش این است که برنامه‌نویسی هم‌روند خیلی وحشتناک نیست، فقط مشکلات و دردسرهای زیادی دارد. اما می‌توانم به شما اطمینان بدهم اگر اصول این کار را بدانید و از خطرات و مشکلات این شیوه‌ی برنامه‌نویسی آگاه باشید، نه تنها ترسناک نیست، بلکه بسیار هم جذاب است. درست مثل بندبازی یا مسابقات اتوموبیل‌رانی سرعت می‌ماند. خطر دارد اما لذت سرعت چیز دیگری است. با استفاده از برنامه‌نویسی موازی هم برنامه‌ی شما واقعاً سریع‌تر می‌شود و هم کاربرپسندی برنامه به میزان بسیار زیادی افزایش پیدا می‌کند. لذا مطمئن باشید از نتیجه‌ی کار حسابی لذت خواهید برد. ابتدا بگذارید نگاهی به سیر تولید پردازنده‌ها و انواع آنها بیندازیم تا ببینیم در چه شرایطی قرار داریم. اطلاع از شرایط موجود اولین کاری است که باید بکنیم.

چند پردازنده‌ها و پردازنده‌های چند هسته‌ای

تا چند سال پیش کامپیوترهای تک پردازنده نظیر پردازنده‌های پنتیوم و آتلون خیلی رواج داشتند. هر بار شرکت‌های اینتل و AMD پردازنده‌ی سریع‌تری بیرون می‌دادند و کاربران به شور و شعف می‌آمدند. می‌توانستید کامپیوتر پنیتوم 3 قدیمی خود را به پنتیوم 4 ارتقاء بدهید و از افزایش سرعت آن واقعاً لذت ببرید. اما چند سالی است که ماجرا تغییر کرده است.

سیر تولید پردازنده‌ها

در سال‌های قبل کورس رقابت بر سر کلاک پردازنده بود و شرکت‌های سازنده‌ی پردازنده مرتباً پردازنده‌های سریع‌تری به بازار می‌دادند. اما کم‌کم متوجه شدند باید سیاست دیگری اتخاذ کنند. چون افزایش سرعت کلاک حرارت زیادی تولید می‌کرد. اگرچه فن‌های قوی‌تر می‌توانستند پردازنده را خنک‌تر کنند، اما فن‌ها نیز بزرگ و بزرگ‌تر می‌شدند. شاید این مساله برای کامپیوترهای رومیزی قابل تحمل باشد اما در کامپیوترهای کیفی چنین چیزی اصلاً شدنی نیست، ضمن این که فن هم باطری مصرف می‌کند. لذا سیاست طراحی پردازنده‌ها تغییر کرد: « اضافه کردن هسته‌های بیشتر ». اما این خبر چندان خوبی برای برنامه‌نویسان معمولی که برنامه‌ها را به شکل عادی (ترتیبی) می‌نوشتند نبود. زیرا در پردازنده‌های چند هسته‌ای برنامه‌ها به شرطی سریع‌تر اجرا می‌شوند که از تمام هسته‌های پردازنده استفاده کنند. به عبارت دیگر به شکل موازی نوشته شوند. چندنخی کردن و موازی نوشتن برنامه هم همان چیزی است که همه احساس بدی نسبت به آن دارند و هر بار به نوعی از زیر بارش در می‌رفتند. اما متاسفانه این بار دیگر نمی‌توانید در بروید. باید مستقیماً با آن روبرو بشوید.

هرب ساتر رئیس استانداردهای ISO زبان C++ که سالها در مورد این زبان و برنامه‌نویسی هم‌روند تجربه و تخصص دارد در سال 2005 در مجله‌ی Dr. Dobb's مقاله‌ای تحت عنوان « The Free Lunch Is Over: A Fundamental Turn Toward Concurrency» نوشت و جامعه‌ی برنامه‌نویسان را به تغییر دیدگاه برنامه‌نویسی برای به دست آوردن حداکثر بهره‌وری از سیستم‌های چند پردازنده‌ی جدید و لزوم این تغییر دیدگاه فراخواند. او توضیح داد برنامه‌نویسان دیگر نمی‌توانند به بهبودی که به صورت مجانی در سرعت پردازنده‌ها توسط شرکت‌های سازنده ایجاد می‌شود اتکا کنند. زیرا تولیدکنندگان پردازنده به جای افزایش فرکانس کلاک، سیاست دیگری در پیش گرفته‌اند (افزایش تعداد هسته‌ها و پردازنده‌ها). مقاله‌ی وی را در آدرس www.gotw.ca/publications/concurrency-ddj.htm می‌توانید مطالعه کنید [1]. معنی تمام این حرف‌ها این است که دیدگاه برنامه‌نویسی معمولی باید تغییر کند و از این به بعد باید موازی فکر کنیم. این همان نقطه‌ای است که اکنون در آن قرار داریم.

[1] نقل از کتاب Professional Parallel Programming with C# 2010 از انتشارات Wrox صفحه‌ی 2.

انواع فناوری پردازنده‌ها

به طور کلی سه نوع فناوری وجود دارد:

- **پردازنده‌های فرانخی (Hyperthreading Processors)**

در این فناوری که به اینتل تعلق دارد پردازنده چند سری رجیستر دارد. لذا می‌تواند خودش را به شکل دو (یا چند) پردازنده نشان بدهد. البته در آن واحد فقط یک سری از رجیسترها را می‌تواند استفاده کند، اما می‌تواند طوری خود را به سیستم عامل و نرم‌افزار نشان بدهد که گویی چندین پردازنده‌ی مجزا است. طبق ادعای اینتل این نوع پردازنده‌ها کارایی را حدود 10 تا 30 درصد بهبود می‌دهند.

- **پردازنده‌های چند هسته‌ای (Multi-Core Processors)**

این نوع پردازنده‌ها چند سالی است که وارد صحنه شده‌اند و در حال حاضر پردازنده‌های دو هسته‌ای، چهار هسته‌ای و گهگاه هشت هسته‌ای در بازار مرسوم هستند. اما اینتل دارد روی پردازنده‌های با تعداد هسته‌های بسیار بیشتر مثلاً 80 هسته کار می‌کند. چنین پردازنده‌ای واقعاً قدرت پردازشی بسیار زیادی خواهد داشت. به این نوع پردازنده‌ها، پردازنده‌های چندین هسته‌ای گفته می‌شود.

- **پردازنده‌های چندین هسته‌ای (Manycore Processors)**

مشابه پردازنده‌های چند هسته‌ای هستند و در آنها نیز پردازنده از چندین هسته تشکیل می‌شود اما فرق بین پردازنده‌ی چند هسته‌ای و چندین هسته‌ای به تعداد و نوع هسته‌های آنها بر می‌گردد. در پردازنده‌ی چندهسته‌ای تعداد هسته‌ها حداکثر 8 تا است و تمام هسته‌ها مشابه هم بوده و همگن (homogenous) هستند اما تعداد هسته‌های پردازنده‌ی چندین هسته‌ای بسیار بیشتر است و هسته‌ها نیز ناهمگن (heterogenous) هستند. این نوع پردازنده‌ها نسل بعدی پردازنده‌ها را تشکیل می‌دهند. در شکل 1-1 نمایی از پنجره‌ی Task Manager در سیستمی با یک پردازنده‌ی چندین هسته‌ای نشان داده شده است[2].

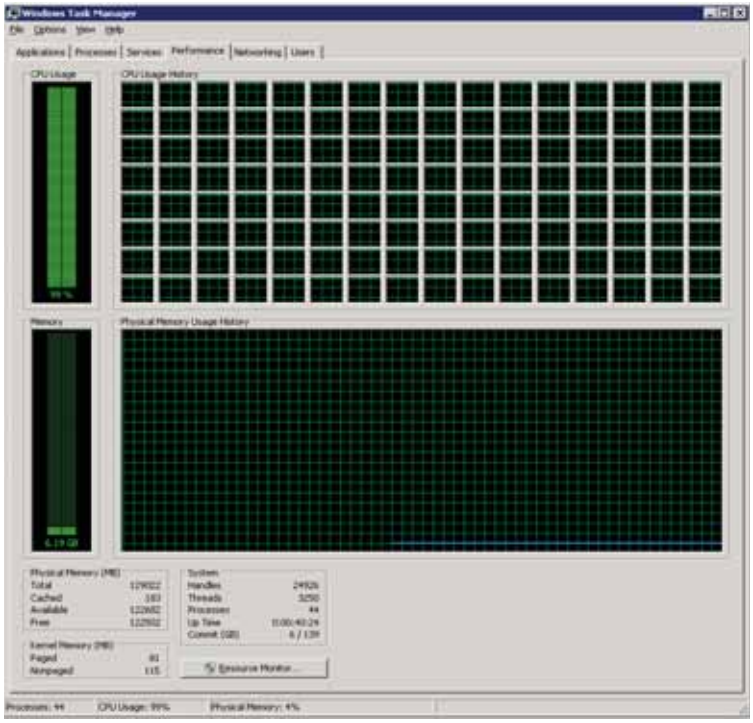
- **چند پردازنده‌ها (MultiProcessors)**

در این فناوری چندین پردازنده بر روی مادربورد گنجانده می‌شود. البته طبعاً مادربورد و کیس نیز بزرگ‌تر شده و در نتیجه مصرف برق آنها بیشتر است و گران‌تر هستند. این نوع سیستم‌ها فعلاً در حوزه‌ی سرویس‌دهنده‌ها رایج هستند و هنوز برای مصارف عمومی و خانگی محبوبیت پیدا نکرده‌اند. در حقیقت شرکت‌های تولید کننده از خودشان می‌پرسند چرا باید کامپیوتری بزرگ، سنگین، گران و پر مصرف تولید کنیم که تازه سرعت برنامه‌ها را عمدتاً به شرطی زیاد می‌کند که برنامه‌ها به شکل موازی نوشده شده باشد. لذا اگر کسی علاقه‌ای به خرید چنین سیستمی نداشته باشد عجیب نیست. اما شکی وجود ندارد که در سال‌های بعد این وضعیت تغییر خواهد کرد.

چیزی که بین تمام این فناوری‌ها مشترک است (البته به جز فناوری فرانخی) این است که در چنین سیستم‌هایی می‌توان برنامه‌ها را به طور *واقعاً موازی* اجرا کرد، نه این که اجرای هم‌روند آنها شبیه‌سازی شود. شبیه‌سازی اجرای

[2] منبع: http://www.danielmoth.com/Blog/windows-7-task-manager-screenshot.aspx

هم‌روند مکانیزمی بود که در سیستم‌های قدیمی و تک‌پردازنده انجام می‌شد. در واقع شما فکر می‌کردید برنامه به طور موازی کار می‌کند، در حالی که واقعاً یک برنامه‌ی ترتیبی بود.



شکل 1-1. پنجره‌ی Task Manager در سیستمی با یک پردازنده‌ی چندین هسته‌ای

وضعیت فعلی و آینده

واقعیت این است که چه دلمان بخواهد چه نخواهد سیر کامپیوترها به سمت چند پردازنده‌ها و پردازنده‌های چند هسته‌ای تغییر کرده و ما هم به عنوان برنامه‌نویس باید دیدگاه‌مان را عوض کنیم. چیزی که می‌توانم به شما بگویم این است که اگر از شیرجه زدن در این استخر واهمه دارید، حداقل در قسمت کم عمق کنار لبه‌ی استخر بنشینید و پایتان را در آب بیندازید. چون به هر حال باید در این استخر شنا کنیم! خبر خوب آن که مایکروسافت با معرفی TPL (کتابخانه‌ی اختصاصی برنامه‌نویسی موازی در NET 4.0.) راه را برای ما کاملاً هموار کرده است.

معماری سیستم‌ها

تک پردازنده‌های چند هسته‌ای

در گذشته شرکت‌های تولید کننده‌ی تک پردازنده‌ها در طراحی پردازنده‌هایشان نکات زیادی در نظر می‌گرفتند تا هم مصرف برق کاهش پیدا کند، هم حرارت کمتری تولید شود و هم قابلیت‌های بهتری برای پردازش موازی فراهم شود تا در نهایت توان عملیاتی پردازنده[3] افزایش پیدا کند. به عنوان مثال امروزه بیشتر پردازنده‌های مدرن قابلیتی دارند که

[3] CPU throughput