میان افزار قابل پیکربندی برای سیستم های زمان-واقعی توزیع شده با وظایف متناوب و غیرمتناوب

چکیده- سیستم های زمان-واقعی توزیع شده مختلف (DRS) باید رویدادهای متناوب و غیرمتناوب را تحت مجموعه ای متناوب از الزامات (شرایط) رفع و رجوع نمایند. در حالیکه میان افزار موجود مانند CORBA زمان-واقعی, به عنوان یک سیستم عامل برای سیستم های توزیع شده با محدودیت های زمانی نویدبخش بوده است, اما فاقد مکانیزم های پیکربندی انعطاف پذیری مورد نیاز برای مدیریت زمانبندی آسان سر هم پیوسته برای گستره وسیعی از DRS مختلف با رویدادهای متناوب و غیرمتناوب است. سهم اولیه این کار, طراحی, پیاده سازی, و ارزیابی عملکرد اولین خدمات میان افزار مولفه قابل پیکربندی برای کنترل پذیرش و تعادل بار رفع و رجوع رویداد متناوب و غیرمتناوب در DRS است. نتایج تجربی, نیاز و اثربخشی رویکرد میان افزار مولفه قابل پیکربندی ما را در حمایت از کاربردهای مختلف با رویدادهای متناوب و غیرمتناوب نشان می دهند و یک سیستم عامل نرم افزار انعطاف پذیر برای DRS با محدودیت های زمانبندی سر هم پیوسته را فراهم می کند.

عبارات شاخص- میان افزار مولفه- تخصیص وظیفه زمان-واقعی دینامیک, متعادل سازی بار و کنترل پذیرش.

1 مقدمه

بسیاری از سیستم های زمان-واقعی توزیع شده (DRS) باید ترکیبی از رویدادهای متناوب و غیرمتناوب, از جمله رویدادهای نامتناوب را با مهلت های سر هم پیوسته رفع و رجوع نمایند که تضمین آنها برای رفتار صحیح سیستم حیاتی است. الزامات (شرایط) برای بهره وری و کیفی نرم افزاری افزوده, استفاده از میان افزار محاسبه شی توزیع شده باز (DOC) مانند CORBA را تحریک می کند. استفاده از میان افزار CORBA در حوزه های DRS مانند هوافضا, مخابرات, سیستم های درمانی, شبیه سازی های تعاملی توزیع شده و تولید یکپارچه-رایانه ای افزایش یافته است که توسط شرایط سخت کیفیت خدمات (QoS) [1] مشخص می شوند. به طور مثال, در یک سیستم نظارت

کارخانه صنعتی, یک هشدار غیرمتناوب را می توان در زمانی که مجموعه ای از قرائت های متناوب سنسور, به معیارهای تشخیص خطر می رسند, تولید نمود. این هشدار باید در پردازنده های متعدد درون یک مهلت سر هم پیوسته پردازش شوند, مثلاً, برای قرار دادن یک فرآیند صنعتی در حالت ایمن از-خرابی. ورودی های کاربر و دیگر قرائت های سنسور می توانند موجب رویدادهای غیرمتناوب زمان-واقعی دیگر شوند.

در حالیکه راه حل های میان افزار زمان-واقعی سنتی مانند Real-Time CORBA [2] and Real-Time Java [3] به عنوان سیستم عامل های نرم افزاری توزیع شده برای سیستم ها با محدودیت های زمانی نویدبخش بوده اند, سیستم های میان افزاری موجود فاقد انعطاف پذیری مورد نیاز برای حمایت از DRS با مفاهیم و شرایط کاربرد مختلف هستند. به طور مثال, متعادل نمودن بار, یک مکانیزم موثر برای رفع و رجوع بارهای کاری زمان-واقعی متغیر در یک DRS است. هرچند, پایداری آن برای DRS عمدتاً وابسته به مفاهیم کاربردی آنهاست. برخی از الگوریتم های کنترل دیجیتال (مثلاً کنترل تناسبی-انتگرالی-مشتق) برای سیستم های فیزیکی, سرراست هستند و ازاینرو برای تخصیص دوباره وظیفه غالب ناشی از تعادل بار, جوابگو هستند, در حالیکه دیگران (مثلاً کنترل تناسبی) دارای چنین محدودیت هایی نیستند. به طور مشابه, گذر از یک کار (گذر از پردازش نمونه های معین یک وظیفه متناوب), برای سرو کار داشتن با بار کاری سیستم گذرا مفید است. در حالیکه گذر از کار, برای کاربردهای کنترل حیاتی معین نامناسب است که در آن از دست دادن کار می تواند نتایج فاجعه باری را در سیستم کنترل شده ایجاد نماید, کاربردهای دیگر اعم از دریافت ویدئو تا ارتباطات می تواند تحمل درجات مختلف گذر از یک کار را میسر سازد [4].

بنابراین, یک چالش کلیدی باز برای DRS, توسعه یک زیرساخت میان افزار انعطاف پذیر است که می توان آن را به راحتی برای حمایت از شرایط مختلف DRS متفاوت پیکربندی نمود. به طور مشخص, خدمات میان افزار مانند توازن بار و کنترل پذیرش باید از انواع راهبردهای جایگزین حمایت نماید (الگوریتم ها و ورودی های متناظر با آن الگوریتم ها). علاوه بر این, پیکربندی این راهبردها باید به گونه ای اصولی و انعطاف پذیر مورد حمایت قرار گیرد, به طوری که توسعه دهندگان سیستم قادر به بررسی پیکربندی های جایگزین باشند, اما بدون انتخاب پیکربندی های نامعتبر ناشی از اشتباه.

بنابراین فراهم نمودن خدمات میان افزار با راهبردهای قابل پیکربندی با چالش های مهم متعدد روبروست: 1) خدمات باید قادر به فراهم نمودن راهبردهای قابل پیکربندی باشند و ابزارهای پیکربندی باید برای میسر نمودن پیکربندی آن راهبردها, اضافه شوند یا گسترش یابند؛ 2) معیارهای خاصل که مشخص می کنند کدام راهبردهای خدمات قابل ترجیح هستند, باید شناسایی شوند و کاربردها باید مطابق با آن معیارها رده بندی شوند؛ و 3) ترکیبات مناسب راهبردهای خدمات باید برای هر رده کاربرد, مطابق با معیارهای مشخصه آن شناسایی شوند. برای پرداختن به این چالش ها, و در نتیجه ارتقای حمایت برای DRS متنوع با رویدادهای متناوب و غیرمتناوب, ما مجموعه ای جدید از خدمات میان افزار مولفه از جمله زمانبندی رویداد سر هم پیوسته, کنترل پذیرش و متعادل سازی بار را طراحی و پیاده سازی نموده است. ما ابزارهای پیکربندی را نیز برای یکپارچه سازی این مولفه های خدمات در هر کاربرد خاص مطابق با معیارهای خاص آن توسعه داده ایم.

بخش های مختلف این تحقیق. در این کار, ما

1. دانش خود از اولین مجموعه خدمات میان افزار مولفه قابل پیکربندی را که از چندین راهبرد کنترل پذیرش و توازن بار برای رفع و رجوع رویدادهای متناوب و غیرمتناوب حمایت می کنند توسعه داده ایم.
2. یک پیش-تجزیه کننده پیکربندی مولفه جدید و واسطه ها را برای پیکربندی کنترل پذیرش زمان-واقعی و خدمات توازن بار را به طور انعطاف پذیر در زمان استقرار سیستم توسعه داده ایم.
3. رده های کاربردهای زمان-واقعی توزیع شده را مطابق با مشخصات خاص تعریف نموده ایم و آنها را به ترکیبات مناسب راهبردها برای خدمات ما مرتبط نموده ایم؛ و
4. یک مطالعه موردی را فراهم نموده ایم که خدمات قابل پیکربندی مختلف را برای یک حوزه با رویدادهای متناوب و نامتناوب به کار می برد. شواهد تجربی از سربارها مرتبط و موازنه ها در میان پیکربندی های خدمات ارائه می دهد و اثربخشی رویکرد ما در آن حوزه را نشان می دهد.

بنابراین, کار ما به طور چشمگیر, قابلیت کاربرد میان افزار زمان-واقعی را به عنوان یک زیرساخت انعطاف پذیر برای DRS ارتقا می بخشد.

بخش 2, سیستم های میان افزار و تئوری زمانبندی که زیربنای رویکرد ماست, معرفی می کند. بخش های 3-5, معماری میان افزار ما, راهبردهای قابل پیکربندی و پیاده سازی های مولفه برای حمایت از هدایت رویداد سر هم پیوسته در DRS ارائه می دهد. بخش 6, گسترس های موتور پیکربندی جدید ما را توصیف می کند که می تواند به طور انعطاف پذیر, راهبردهای مختلف را برای خدمات ما مطابق با هر شرایط کاربرد پیکربندی نماید. بخش 7, عملکرد رویکرد ما, از جمله موازنه ها مان ترکیبات راهبرد خدمات مختلف را ارزیابی نماید و سربارهای معرفی شده توسط رویکرد ما را مشخص نماید. بخش 8, یک نظرسنجی از کار مرتبط را ارائه می دهد و ما نتایج را در بخش 9 ارائه می دهیم.

2 پیش زمینه

مدل وظیفه, ما DRS متشکل از سیستم های فیزیکی تولیدکننده رویدادهای متناوب و نامتناوب را در نظر می گیریم که باید در سیستم عامل های محاسبه توزیع شده تحت مهلت های سر هم پیوسته پردازش شوند. از اینرو, پردازش یک دنباله از رویدادهای مرتبط, یک وظیفه نامیده می شود. یک وظیفه Ti از زنجیره ای وظایف فرعی $T_{i,j}(1 \leq j \leq n_i)$ واقع در پردازنده های مختلف تشکیل شده است. اولین وظیفه فرعی Ti,1 از یک وظیفه Ti, توسط یک رویداد تایمر متناوب یا یک رویداد نامتناوب تولید شده تحریک می شود. به محض تکمیل, یک وظیفه فرعی Ti,j, رویداد دیگری را باعث می شود که موجب تحریک وظیفه فرعی بعدی آن, $T_{i,j+1}$ می شود. هر وظیفه فرعی از یک وظیفه متناوب, یک دنباله از کارهای فرعی است. هر وظیفه متناوب, یک دنباله از کارها است که هر کار, زنجیره ای از کارهای فرعی از هر یک از وظایف فرعی این وظیفه است. زمان ورود یک کار یا کار فرعی, زمانیست که برای اجرا دردسترس قرار می گیرد. زمان انتشار یک کار یا کار فرعی بعد از ورود آن رخ دهد, بعد از پذیرش آن توسط کنترل کننده پذیرش, زمانی که توسط سیستم, برای اجرا آزاد می شود. هر کار از یک وظیفه باید ظرف یک مهلت سر هم پیوسته تکمیل شود که ماکزیمم زمان پاسخ مجاز آن است. دوره یک وظیفه متناوب, زمان بین-ورود کارهای فرعی متوالی از اولین وظیفه فرعی از وظیفه متناوب است. یک وظیفه غیرمتناوب, دوره تناوب ندارد. زمان میان-ورودی بین کارهای فرعی متوالی از اولین وظیفه فرعی آن می تواند به طور گسترده تغییر نماید و به طور خاصل می

تواند کمی دلخواهانه باشد. زمان اجرای بدترین مورد از هر وظیفه فرعی, مهلت سر هم پیوسته از هر وظیفه و دوره تناوب هر وظیفه متناوب در سیستم, شناخته شده هستند.

میان افزار مولفه. سیستم عامل های میان افزار مولفه, روشی موثر برای دستیابی به استفاده مجدد از مصنوعات نرم افزاری به صورت سفارشی است. در این سیستم عامل ها, مولفه ها, واحدهای پیاده سازی و ترکیبی هستند که با مولفه های دیگر توسط پورت ها همکاری می کنند. پورت ها, زمینه های مولفه ها را از پیاده سازی های واقعی آن جدا می سازند. سیستم عامل های میان افزار مولفه, محیط های اجرا و خدمات رایج را فراهم می سازند و از ابزارهای اضافی برای پیکربندی و به کارگیری مولفه ها حمایت می کنند. در کار قبلی, ما اولین نمونه برداری یک خدمت کنترل پذیرش میان افزار را توسعه دادیم که از رویدادهای متناوب و نامتناوب حمایت می کنند [5] (در TAO, یک میان افزار Real-Time CORBA استفاده شده گسترده). هرچند, خدمت کنترل پذیرش قبلی ما, شامل مجموعه ای ثابت از راهبردها می شود. همانطور که در بخش 4 نشان داده شده است, یک مجموعه قابل پیکربندی و متنوع از خدمات میان-عملیاتی و راهبردهای عملیاتی بر حمایت از DRS با مفاهیم کاربرد مختلف مورد نیاز است. متاسفانه, گسترش پیاده سازی ها که به طور مستقیم متکی بر میان افزار شی توزیع شده است, مشکل می باشد, مانند خدمت کنترل پذیرش اصلی ما. به طور خاص, در سیستم های میان افزار در حال تغییر, راهبرد حمایت شده, به تغییرات صریح برای کد خدمت نیاز دارد که می تواند در عمل خسته کننده و مستعد در خطا باشد.

Component-Integrated ACE ORB (CIAO) [6], Light Weight CORBA Component Model (CCM) را پیاده سازی می کند [7] و در بالای TAO [8] بروکر درخواست شی CORBA زمان-واقعی (ORB) ساخته می شود. CIAO, خط مشی های زمان واقعی رایج را به صورت واحدهای قابل نصب و قابل پیکربندی چکیده می نماید. هرچند, CIAO از زمانبندی وظیفه نامتناوب, کنترل پذیرش یا توازن بار حمایت نمی کند. برای توسعه یک زیرساخت انعطاف پذیر برای DRS, در این کار, ما کنترل پذیرش جدید و خدمات توازن بار, را هر یک با مجموعه ای راهبردهای خدمات جایگزین در بالای CIAO توسعه می دهیم. علاوه براین, ما CIAO را برای پیکربندی گسترش دادیم و هر دو خدمات را مدیریت نمودیم.

DAnCE [9], یک موتور پیکربندی و استقرار مولفه QoS-محور است که Object Management Group (OMG)'s Light Weight CCM Deployment و مشخصات پیکربندی [7] را پیاده سازی می نماید. DAnCE, توصیفات پیکربندی/استقرار مولفه را تجزیه می کند و به طور خودکار ORBها, کانتینرها و منابع سرور را در زمان راه اندازی سیستم, برای تقویت شرایط QoS سر هم پیوسته پیکربندی و به کارگیری می نماید. هرچند, DAnCE, ویژگی های ضروری معین مورد نیاز برای پیکربندی کنترل پذیرش و خدمات توازن بار را به درستی فراهم نمی کند, مثلاً ترکیبات نامعتبر راهبردهای خدمات ما را مجاز نمی سازد.

زمانبندی نامتناوب. وظیفه نامتناوب به طور گسترده در تئوری زمانبندی زمان-واقعی, از جمله کار روی سرورهای نامتناوب مطالعه شده است که زمانبندی وظایف متناوب و نامتناوب را ادغام می نمایند [10]. آزمون های قابلیت زمانبندی جدید بر اساس مرزهای استفاده نامتناوب (AUBها) [11] و یک رویکرد کنترل پذیرش جدید [12] نیز اخیراً معرفی شد. در کار قبلی ما [5], ما خدمات کنترل پذیرش را برای دو تکنیک زمانبندی نامتناوب مناسب (مرز استفاده نامتناوب [11] و سرور قابل قابل تاخیر [13]) در TAO پیاده سازی و ارزیابی نمودیم. چون AUB دارای عملکرد قابل مقایسه با سرور قابل تاخیر است و به مکانیزم های زمانبندی با پیچیدگی کمتر در میان افزار نیاز دارد, ما به طور انحصاری روی تکنیک AUB در این مقاله تمرکز می نماییم. تجربیات ما با AUB گزارش شده در این مقاله, نشان می دهد که چگونه قابلیت پیکربندی دیگر تکنیک ها می توانند درون میان افزار مولفه زمان واقعی, به روشی مشابه ادغام شوند.

با رویکرد AUB, سه نوع راهبرد خدمت باید برای فراهم نمودن حمایت اصولی و انعطاف پذیر برای DRS متنوع با وظایف متناوب و نامتناوب قابل پیکربندی شوند: 1) زمانی که قابلیت پذیرش ارزیابی می شود (برای موازنه بین ذره ذره بودن, و بنابراین, بدبینی تضمین های پذیرش), 2) زمانی که سهم وظایف فرعی کامل شده را بتوان از تحلیل قابلیت زمانبندی استفاده شده برای کنترل پذیرش حذف نمود (برای بهبود دقت تحلیل قابلیت زمانبندی و در نتیجه کاهش انکارهای بدبینانه وظایف عملی), 3) زمانی که کارهای وظایف را بتوان به پردازنده های مختلف منسوب نمود (برای متعادل نمودن بار و بهبود عملکرد سیستم).

در AUB [11], مجموعه وظایف کنونی S(t) در هر زمان t, به صورت مجموعه ای از وظایف تعریف می شود که $S(t) = \{T_i \mid A_i \leq t < A_i + D_i\}$, که در آن Ai, زمان آزاد سازی کار فرعی اول از کار کنونی برای وظیفه Ti است و کارهای را آزاد نموده اند اما مهلت های آنها منقضی نشده است. از اینرو, Di مهلت نسبی کار کنونی وظیفه Ti است. استفاده مصنوعی از پردازنده j در هر زمان t, Uj(t), به عنوان مجموع کاربردهای وظیفه فرعی منفرد در پردازنده تعریف می شود که روی تمام وظایف کنونی افزوده می شود. مطابق با تحلیل AUB, یک سیستم, بالاترین مرز مطلوبیت مصنوعی قابل زمانبندی خود را تحت الگوریتم زمانبندی مونوتونیک مهلت سر هم پیوسته (EDMS) تحت فرضیات معین به دست می آورد. تحت EDMS, یک وظیفه فرعی دارای اولویت بالاتر است, اگر به یک وظیفه با یک مهلت سر هم پیوسته کوتاهتر تعلق داشته باشد. توجه داشته باشید که AUB وظایف نامتناوب را از متناوب متمایز می کند. تمام وظایف با استفاده از خط مشی زمانبندی یکسان زمانبندی می شوند. تحت EDMS, وظیفه Ti, در صورتی مهلت خود را رعایت می کند که شرط قابلیت زمانبندی بعدی برقرار باشد [11]:

$$\sum_{j=1}^{n_i} \frac{U_{V_{ij}}(1 - U_{V_{ij}}/2)}{1 - U_{V_{ij}}} \leq 1, \quad (1)$$

که در آن Vij, پردازنده jام است که وظیفه Ti را بازدید می کند. یک وظیفه (یا یک کار فردی) را زمانی می توان پذیرش نمود که این شرط همچنان برای تمام وظایف پذیرش شده کنونی و این وظیفه برآورده شود. چون کاربردها, گذر از کار را تحمل می کنند یا نمی کنند, خواه این شرط تنها زمانی چک شود که اولین کار یک وظیفه فرا رسد یا زمانی که هر کار سر برسد, باید قابل پیکربندی باشد.

مطابق با تعریف مجموعه وظیفه کنونی در AUB, یک وظیفه در مجموعه وظیفه کنونی باقی می ماند, حتی اگر کامل شده باشد, تا زمانی که مهلت آن منقضی نشده باشد. برای کاهش بدبینی تحلیل AUB, یک قاعده تنظیم دوباره, در [11] معرفی می شود. زمانی که یک پردازنده, بیکار می شود, سهم کارهای فرعی کامل شده برای استفاده مصنوعی از پردازنده را می توان بدون تاثیرگذاری بر صحت وضعیت قابلیت زمانبندی حذف نمود (نامعادله 1). از

زمانی که قاعده تنظیم دوباره, سربار اضافی را معرفی می کند, خواه سهم کارهای فرعی نامتناوب کامل شده یا سهم کارهای فرعی متناوب و نامتناوب کامل شده قابل حذف باشند یا نباشند, ابتدا باید قابل پیکربندی باشد. تحت تحلیل قابلیت زمانبندی مبتنی بر AUB, توازن بار نیز می تواند به طور موثر, عملکرد سیستم را بهبود بخشد [11]. هرچند, برخی از کاربردها به ذخیره سازی حالت پایدار بین کارهای یک وظیفه نیاز دارند, بنابراین باید قابل پیکربندی شود, خواه یک وظیفه قابل تخصیص به یک پردازنده متفاوت برای هر کار باشد و خواه نباشد.

3 معماری میان افزار

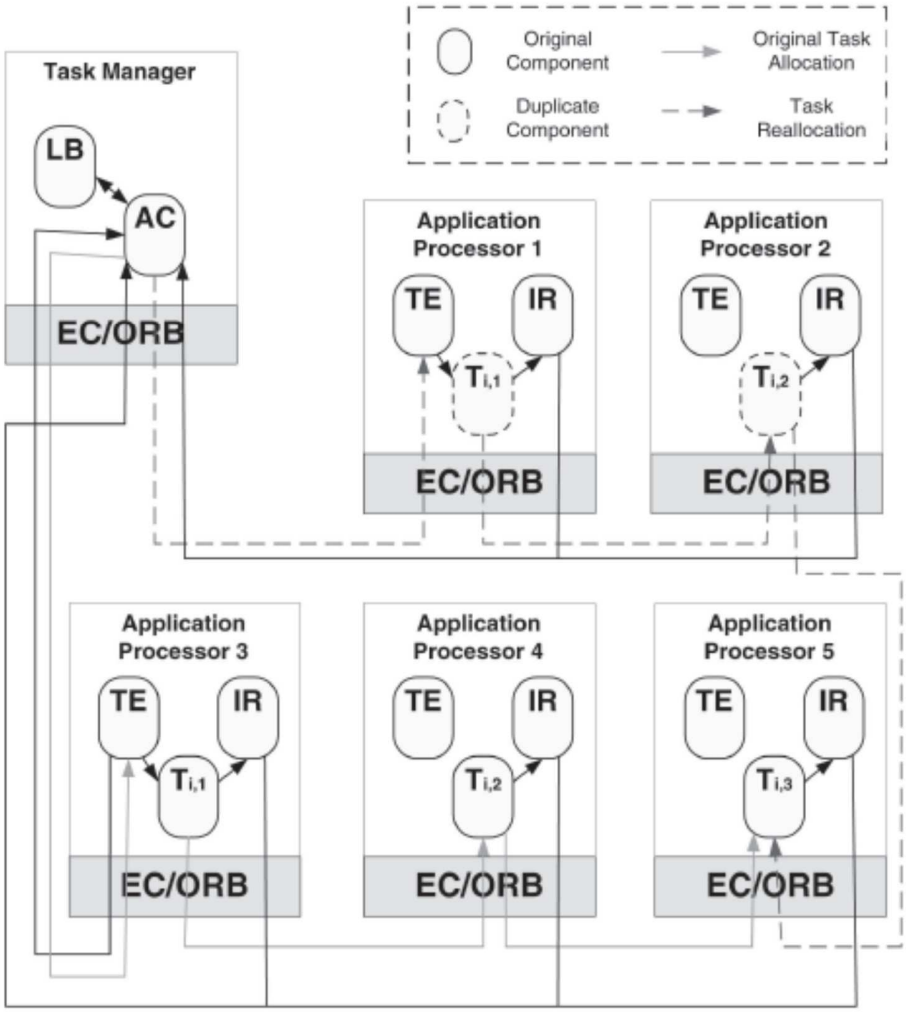
برای حمایت از وظایف متناوب و نامتناوب سر هم پیوسته در کاربردهای زمان-واقعی توزیع شده مختلف, ما یک معماری میان افزار جدید را توسعه داده ایم, ویژگی کلیدی رویکرد ما, یک چارچوب مولفه قابل پیکربندی است که می توان آن را برای مجموعه های مختلف از وظایف متناوب و نامتناوب سفارشی نمود. چارچوب ما, مولفه های کنترل کننده پذیرش قابل پیکربندی (AC), متوازن کننده بار (LB), و تنظیم کننده دوباره بیکار (IR) را فراهم می کند که با مولفه های کاربرد و مولفه های تاثیرگذار وظیفه (TE) تعامل پیدا می کنند. مولفه AC, کنترل پذیرش آنلاین و آزمون های قابلیت مقیاس بندی را برای وظایفی فراهم می کند که به طور دینامیک در زمان اجرا سر می رسند. مولفه LB, یک برنامه انتساب وظیفه قابل قبول را در صورتی برای کنترل کننده پذیرش فراهم می کند که وظیفه ورود جدید قابل پذیرش باشد. هر مولفه IR, تمام کارهای تکمیل شده فرعی در یک پردازنده را زمانی به مولفه AC گزارش می دهد که پردازنده بیکار می شود, بنابراین مولفه AC می تواند سهم خود را از مطلوبیت مصنوعی محاسبه شده حذف نماید و مطابق با قاعده تنظیم دوباره بیکاری, بدبینی تحلیل AUB را در زمان اجرا کاهش دهد. در هر پردازنده, یک مولفه TE, زمانی به مولفه AC اخطار می دهد که کارهای جدید فرا رسد و کارهای پذیرش شده آزاد شوند.

شکل 1, معماری میان افزار توزیع شده ما را توضیح می دهد. تمام پردازنده ها توسط TAO ORB's federated Event Channel (EC) [14] متصل می شوند که توسط EC/ORB در شکل 1 نشان داده می شوند. فلش های سیاه در شکل 1, نشاندهنده یک رویداد EC تحریک شده یا یک فراخوان روش ORB فرستاده شده است. EC, رویدادها را از طریق

کانال های رویداد موضعی, دروازه راه ها و کانال های رویداد از راه دور به مصرف کنندگان رویداد نشسته روی پردازنده های مختلف تحریک می کند. ما یک مولفه AC و یک مولفه LB را به کارگیری می نماییم که با هم برای انجام مدیریت وظیفه روی یک پردازنده و یک مولفه IR و یک مولفه TE در هر یک از پردازنده های کاربرد متعدد کار می کنند.

شکل 1, نشاندهنده یک مثال از Ti وظیفه سر هم پیوسته نمونه متشکل از سه وظیفه فرعی متوالی $T_{i,1}, T_{i,2}$ و $T_{i,3}$ در حال اجرا روی پردازنده های جداگانه است. $T_{i,1}$ و $T_{i,2}$ دارای نسخه هایی در دیگر پردازنده های برنامه هستند. یک مولفه اصلی و نسخه های آن, مولفه های جایگزین برنامه هستند که می توانند یک وظیفه فرعی را با انتساب وظیفه فرعی واقعی تعیین شده توسط مولفه LB در زمان اجرا, به انجام برسانند. به خاطر این بحث, فرض کنید که وظیفه Ti, در پردازنده برنامه 3 می رسد, مولفه TE روی آن پردازنده, یک رویداد "ورود وظیفه" را برای مولفه AC تحریک می کند و تا زمانی این وظیفه برقرار است که یک رویداد "قبول" را از مولفه AC دریافت کند. مولفه AC و مولفه LB, تضمیم می گیرند که آیا این وظیفه را بپذیرند یا خیر و اگر اینگونه باشد, وظایف فرعی آن را به کجا منسوب نمایند. خطوط توپر و خطوط تکه تکه نشاندهنده دو انتساب ممکن وظایف فرعی است. اگر اولین وظیفه فرعی Ti,1, به پردازنده ای منسوب نشود که Ti به آن می رسد, ما این انتساب را یک تخصیص دوباره وظیفه می نامیم.

یک مزیت این معماری AC/LB متمرکز اینست که به همزمان سازی در میان کنترلرهای پذیرش توزیع شده نیاز ندارد. در مقابل, در یک معماری مدیریت وظیفه توزیع شده, مولفه های AC در چندین پردازنده ممکن است به هماهنگ و همزمان سازی با یکدیگر نیاز داشته باشد تا تصمیمات درست گرفته شود, زیرا پذیرش یک وظیفه سر هم پیوسته می تواند بر قابلیت زمانبندی وظایف دیگر واقع شده در چندین پردازنده متاثر, تاثیر بگذارد.



شکل 1. معماری میان افزار مولفه: فلش های سیاه نشاندهنده یک تحریک رویداد یا فراخوان روش است؛ مولفه های اصلی و دوبل, جایگزین های اجزای یک وظیفه فرعی هستند؛ فرض کنید وظیفه Ti در پردازنده برنامه 3 می رسد.

یک عیب بالقوه معماری متمرکز اینست که مولفه AC می تواند یک تنگنا باشد و بنابراین بر قابلیت مقیاس پذیری تاثیر بگذارد. هرچند, زمان محاسبه تحلیل قابلیت مقیاس پذیری به طور چشمگیری کمتر از زمان های اجرای وظیفه در چندین DRS است که محدودیت های قابلیت مقیاس پذیری یک راه حل متمرکز [5] را کاهش می دهد. مدیریت وظیفه متمرکز نیز می تواند یک نقطه خرابی تک باشد که به طور منفی بر موجودیت و قابلیت بقای سیستم تاثیر می گذارد. کنترل پذیرش و توازن بار می تواند با استفاده از تکنیک های تحمل-خطای منفعل و فعال موجود برای سیستم های زمان-واقعی تکرار شود [16] ,[15]. هرچند, پرداختن به مجموعه ای کامل از موضوعات مدیریت

وظیفه تحمل کننده-خطا, فراتر از محدوده این مقاله است و به عنوان یک مبحث بالقوه آینده از این کار, کنار گذاشته می شود. به طور خلاصه, در حالیکه رویکرد میان افزار مولفه زمان-واقعی ما می تواند برای استفاده از یک معماری مدیریت وظیفه توزیع شده تر گسترش یابد, ما یک رویکرد متمرکز را با پیچیدگی و سربار کمتر اتخاذ نموده ایم که به ما اجازه می دهد تا بر کسب انعطاف پذیری سیستم از طریق خدمات میان افزار مولفه تمرکز نماییم.

4 نگاشت مشخصات DRS به راهبردهای میان افزار

سهم کلیدی این مقاله, رده بندی مشخصاتی است که در مجموعه نماینده معقول کاربردهای DRS مشترک هستند و نگاشت آنها به راهبردهای خدمات مناسب میان افزار. در این بخش, مجموعه ای از معیارهای استفاده شده برای رده بندی مشخصات DRS را ارائه می دهیم و نحوه نگاشت این معیارها به راهبردهای خدمات مختلف حمایت شده توسط میان افزار خود را تحلیل می نماییم.

جدول 1معیارها و راهبردهای خدمت میان افزار

	No	Yes	Some
C1:Job Skipping	AC per Task	AC per Job	
C2:Overhead Tolerance	No IR	IR per Job	IR per Task
C3:State Persistency	LB per Job	LB per Task	
C4:Component Replication	No LB	LB	

4.1 معیارهای DRS

ما از چهار معیار برای متمایز نمودن نحوه حمایت DRS با وظایف متناوب و نامتناوب استفاده می کنیم: گذر از کار (معیار C1)؛ تحمل سربار (معیار C2)؛ پایداری حالت (معیار C3)؛ و تکرار مولفه (معیار C4).

گذر از کار بدان معنیست که برخی کارها از یک وظیفه اجرا شوند در حالیکه دیگر کارها از همان وظیفه ممکن است پذیرفته نشوند. برخی برنامه های, مانند پخش ویدئو و دیگر اشکال حس کردن (سنسور) که تلفات را تحمل می کنند, می توانند گذر از کار را تحمل نمایند, در حالیکه در برنامه های کنترل بحرانی, زمانی که یک وظیفه پذیرفته می شود, تمام کارهای آن باید مجاز به اجرا باشند.

تحمل سربار وابسته به محدودیت های سربار خاص یک برنامه است: ما منابع مختلف سربار را برای خدمات خود در بخش 7.3 مشخص می کنیم, به طوری که توسعه دهندگان هر برنامه می توانند تصمیم بگیرند که آیا آن سربارها در صورت کاربرد برای قابلیت زمانبندی بهبودیافته, بیش از حد خواهند بود یا قابل قبول.

پایداری حالت بدان معنیست که حالات برای ذخیره بین کارهای یک وظیفه نیاز می شوند. برای سیستم های کنترل تناسبی [17], وظیفه, بدون حالت است و فقط به اطلاعات کنونی نیاز دارد, بنابراین کارها را می توان به طور دینامیکی دوباره اختصاص داد. هرچند, برای سیستم های کنترل انتگرال [17], وظایف به محاسبه افزایشی نیاز دارند و برای تخصیص دوباره شغل مناسب نیستند.

تکرار مولفه وابسته به شرایط توان عملیاتی کاربرد است. تکرار در اینجا برای کاهش تاثیر از طریق توزیع بار استفاده می شود, نه برای مقاصد تحمل-خطا. تنها آن دسته از برنامه های دارای مولفه های تکراری می توانند از تخصیص دوباره وظیفه حمایت نمایند, در حالیکه آنهایی که نمی توانند تکرار شوند (مثلاً به دلیل محدودیت ها روی موقعیت سنسورها و محرک ها) نمی توانند از تخصیص دوباره وظیفه حمایت نمایند.

مطابق با این معیارهای کاربرد مختلف, مولفه های AC, IR, و LB را می توان برای استفاده از راهبردهای مختلف پیکربندی نمود. اینکه برای هر مولفه, کدام راهبرد مناسب تر است, وابسته به این معیارهاست. جدول 1 نشاندهنده اینست که چگونه این معیارها به طبقه بندی برنامه های DRS کمک می کنند, که به نوبه خود, انتخاب راهبردهای خدمت میان افزار متناظر را میسر می سازد. ما تمام راهبردها را با صفات قابل پیکربندی متناظر طراحی نموده ایم و یک پیش-تجزیه کننده پیکربندی را و یک واسطه پیکربندی مولفه را فراهم می کنیم (که در بخش 6 توصیف شده است) تا به توسعه دهندگان اجازه دهیم تا هر خدمت را به طور انعطاف پذیری مطابق با هر نیاز خاص کاربرد انتخاب و پیکربندی نمایند. ما در حال حاضر, راهبردهای مختلف برای هر مولفه و موازنه در میان آنها را بررسی می نماییم.

4.2 راهبردهای کنترل پذیرش (AC)

کنترل پذیری, مزایای چشمگیر را برای سیستم ها بدون وظایف متناوب و نامتناوب, با فراهم نمودن تضمین های قابلیت زمانبندی آنلاین برای وظایف ورودی به طور دینامیک ارائه می دهد.

مولفه AC ما از دو راهبرد متفاوت حمایت می کند: AC در هر وظیفه و AC در هر کار. AC در هر وظیفه, آزمون پذیرش را فقط زمانی انجام می دهد که یک وظیفه برای اولین بار فرا می رسد, در حالیکه AC در هر کار, آزمون پذیرش را زمانی انجام می دهد که یک کار از یک وظیفه می رسد. تنها برنامه های برآورده کننده معیار C1, برای راهبرد دوم مناسب هستند, زیرا نمی تواند برخی کارهای را پذیرش نماید. علاوه بر این, راهبرد دوم, بدبینی را کاهش می دهد اما با افزایش سربار. بنابراین, توسعه دهنده برنامه باید موازنه ها بین سربار و بدبینی در انتخاب یک پیکربندی مناسب را در نظر گیرد.

AC در هر وظیفه. با در نظر گرفتن سربار پذیرش و زمان های میان-ورود تثبیت شده از وظایف متناوب, یک راهبرد, انجام یک آزمون پذیرش, فقط در زمانیست که یک وظیفه متناوب برای اولین بار می رسد. زمانی که یک وظیفه متناوب از آزمون پذیرش قبول می شود, تمام کارهای آن, زمانی که می رسند, مجاز به آزادسازی فوری می شوند. این راهبرد موجب بهبود بازده میان افزار می شود اما با افزایش بدبینی آزمون پذیرش. در تحلیل AUB [11], سهم یک کار برای مطلوبیت مصنوعی یک پردازنده را زمانی می توان حذف نمود که مهلت کار منقضی می شود (یا زمانی که CPU بیکار می شود, اگر قاعده راه اندازه دوباره کننده استفاده شود و کار فرعی کامل شده باشد). با این حال, اگر کنترل پذیرش تنها در زمان رسیدن وظیفه انجام شود, مولفه AC باید, کاربرد مصنوعی وظیفه را در سراسر طول عمر آن رزرو نماید. در نتیجه, نمی تواند کاربرد مصنوعی بین مهلت یک کار و رسیدن کار بعدی از همان وظیفه را کاهش می دهد و این به تصمیمات پذیرش بدبینانه منجر می شود [11].

AC در هر کار. اگر گذر از یک کار در یک وظیفه متناوب ممکن باشد (معیار C1), راهبرد جایگزین برای کاهش بدبینی, اعمال آزمون پذیرش برای هر کار از یک وظیفه متناوب است. این راهبرد برای سیستم های مختلف عملی است, زیرا آزمون AUB عمدتاً زمانی کارایی دارد که برای AC استفاده شود, همانطور که در بخش 7.3 توسط اندازه گیری های سربار ما نشان داده شده است.

4.3 راهبردهای تنظیم دوباره بیکار (IR)

بدون قاعده تنظیم دوباره AUB, یک کار در مجموعه کنونی باقی می ماند حتی اگر کامل شده باشد, تا زمانی که مهلت آن منقضی نشده باشد. بنابراین, استفاده از قاعده تنظیم دوباره می تواند سهم کارهای فرعی کامل شده را زودتر از حذف نماید که بدبینی آزمون قابلیت زمانبندی AUB را کاهش می یابد [11] ,[5]. سه راهبرد برای پیکربندی مولفه های IR در رویکرد ما وجود دارد که مطابق با یک تحمل سربار کاربرد (معیار C2) است. سه راهبرد اول از سربار تنظیم دوباره جلوگیری می کند, اما بدبینانه ترین است. راهبرد سوم, سهم کارهای متناوب و نامتناوب تکمیل شده را به طور غالب تر از دو راهبرد دیگر حذف می کند. هرچند دارای حداقل بدبینی است, بیشترین سربار را ایجاد می کند. دومین راهبرد, یک موازنه بین اولین و سومین راهبرد را ارائه می دهد.

بدون IR. اولین راهبرد, استفاده از عدم تنظیم دوباره است, به طوری که اگر کارهای فرعی, اجراهای خود را تکمیل نمایند, سهم کارهای فرعی تکمیل شده برای کاربرد مصنوعی پردازنده تا زمان مهلت کار حذف نمی شود. این راهبرد از سربار تنظیم دوباره جلوگیری می کند, اما بدبینی تحلیل قابلیت مقیاس بندی را افزایش می دهد.

IR در هر وظیفه. دومین راهبرد اینست که هر مولفه IR, کارهای فرعی نامتناوب تکمیل شده را در یک پردازنده ثبت می کند. زمانی که پردازنده, بیکار است, یک رشته با پایین ترین اولویت که یک آشکارساز بیکار نامیده می شود, شروع به اجرا می کند و کارهای فرعی نامتناوب کامل شده را به مولفه AC از طریق یک رویداد تنظیم دوباره حالت بیمار گزارش می دهد. برای اجتناب از گزارش مکرر, آشکارساز بیکار تنها زمانی گزارش می دهد که یک کار فرعی نامتناوب تکمیل شده جدید وجود داشته باشد که مهلت آن منقضی نشده باشد.

IR در هر شغل. سومین راهبرد اینست که هر مولفه IR, نه تنها کارهای فرعی نامتناوب تکمیل شده را ثبت می کند و گزارش می دهد, بلکه کارهای فرعی تکمیل شده از وظایف فرعی متناوب را ثبت می کند و گزارش می دهد.

4.4 راهبردهای توازن بار (LB)

تحت AC مبتنی بر-AUB, توازن بار به طور موثر می تواند عملکرد سیستم را در رویارویی با ورود وظیفه دینامیک بهبود بخشد [11]. ما از یک الگوریتم ابتکاری برای انتساب وظایف فرعی به پردازنده ها در زمان اجرا استفاده می کنیم که همیشه یک وظیفه فرعی را به پردازنده با کمترین کاربرد مصنوعی در میان تمام پردازنده هایی که بر اساس

آن, مولفه کاربرد متناظر با وظیفه تکرار شده است, منسوب می نماید (معیار C4). از آنجا که حرکت یک وظیفه فرعی بین پردازنده ها, سربار اضافی را ایجاد می کند, زمانی که ما یک وظیفه جدید را می پذیریم, تنها انتساب آن وظیفه جدید را تعیین می کنیم و برنامه انتساب برای هر وظیفه دیگر را در مجموعه وظیفه کنونی تغییر نمی دهیم. این خدمت نیز دارای سه راهبرد است. اولین راهبرد برای کاربردهایی مناسب است که نمی توانند معیار C4 را برآورده سازند. دومین راهبرد, برای کاربردهایی دارای بیشتری کاربرد است که نمی تواند معیار C3 و C4 را برآورده سازد. سومین راهبرد, برای کاربردهایی, مناسب ترین است که تنها C4 را برآورده می سازند, اما نمی تواند معیار C3 را برآورده سازد.

عدم LB. این راهبرد, توازن بار را انجام نمی دهد. هر وظیفه فرعی دارای یک نسخه تکراری است و به یک پردازنده خاص منسوب می شود.

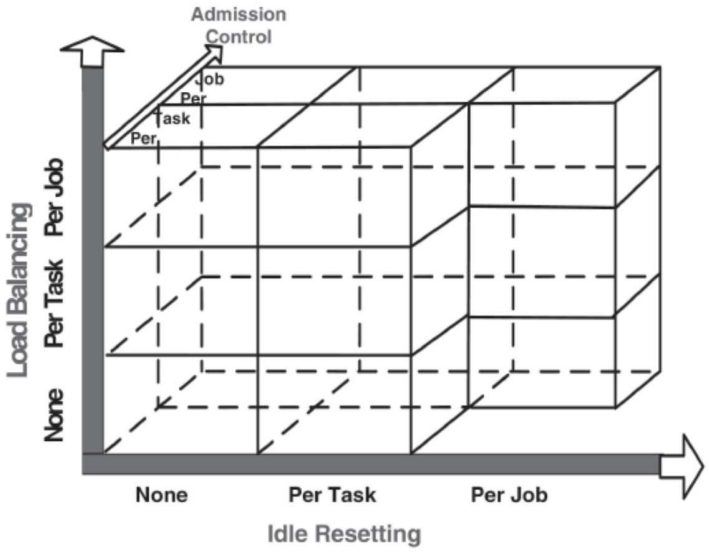
LB در هر وظیفه. هر وظیفه تنها یک بار, در زمان ورود اولیه آن منسوب خواهد شد. این راهبرد برای کاربردهایی مناسب است که باید حالت پایدار را بین هر دو کار متوالی از یک وظیفه متناوب حفظ نماید.

LB در هر کار. سومین راهبرد, انعطاف پذیرترین است. تمام کارها, زمانی که می رسند, از یک وظیفه متناوب مجاز به انتساب به پردازنده های مختلف هستند.

4.5 ترکیب راهبردهای AC, IR و LB

زمانی که ما از مولفه های AC, IR و LB با هم استفاده می کنیم, راهبردهای آنها می توانند در 18 ترکیب مختلف پیکربندی شوند. هرچند, برخی ترکیبات راهبردها نامعتبر هستند. ترکیب AC در هر وظیفه/ IR در هر کار, معقول نیست, زیرا تنظیم دوباره حالت بیکار در هر کار به معنی کاربردهای مصنوعی تمام کارهای فرعی تکمیل شده وظایف فرعی متناوب باید از کنترلر پذیرش مرکزی حذف شوند, اما کنترل پذیرش در هر وظیفه نیاز به این دارد که کنترلر پذیرش, کاربرد مصنوعی برای تمام وظایف متناوب پذیرفته شده را رزرو نماید, بنابراین, یک وظیفه متناوب پذیرفته شده نباید از طریق کنترل پذیرش, دوباره قبل از آزادسازی کارهای آن پیش برود. بنابراین این دو الزام متناقض هستند و می توانیم استثنا قائل شویم که این پیکربندی های متناظر, نامعتبر هستند. حذف این ترکیب نامعتبر

AC/IR به معنی حذف 3 ترکیب AC/Ir/LB نامعتبر است, بنابراین تنها 15 ترکیب معقول از راهبردهای باقیمانده وجود دارد. با این درجه از پیچیدگی در گرفتن تصمیمات درست طراحی پیکربندی صحیح, یک توسعه دهنده برنامه از حمایت شناختی در پیکربندی راهبردهای مختلف منفعت خواهد برد. یک مزیت معماری میان افزار ما, و موتور پیکربندی اینست که آنها به توسعه دهندگان برنامه اجازه می دهند تا خدمات میان افزار را برای کسب هر ترکیب معتبر از راهبردها پیکربندی نمایند و در عین حال ترکیبات نامعتبر را که در بخش 6 بررسی نمودیم مجاز نمی سازند.



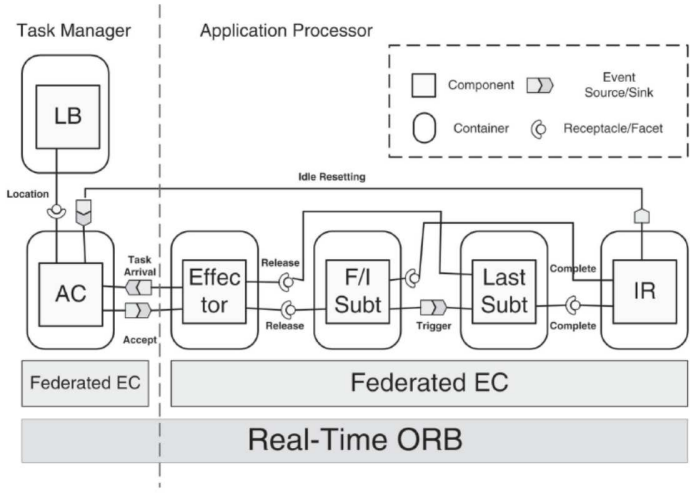
شکل 2. ابعاد راهبرد خدمات میان افزار

همانطور که شکل 2 نشان می دهد, گزینه های پیکربندی را می توان در محورهای قابلیت پیکربندی راهبرد برای هر یک از سه خدمات میان افزار تقسیم بندی نمود: کنترل پذیرش, تنظیم دوباره حالت بیکار و توازن باز, گزینه های پیکربندی مختلف در هر یک از سه محور و تاثیری که آنها ممکن است داشته باشند و پیکربندهای متناقض, به طور کلی معین می شوند و همانطور که در بخش 6 بررسی نمودیم, مبنای حمایت خودکار از توسعه دهندگان برنامه در پیکربندی خدماتی که میان افزار ما فراهم می کند را تشکیل می دهند.

5 پیاده سازی مولفه

استانداردهای میان افزار مولفه قابل پیکربندی مانند CCM [18] می توانند با تعریف یک الگوی برنامه نویسی مبتنی بر مولفه, به کاهش پیچیدگی توسعه DRS کمک نمایند. آنها همچنین با تعریف یک چارچوب پیکربندی استاندارد برای بسته بندی و به کارگیری مولفه های نرم افزاری قابل استفاده مجدد, مفید خواهند بود. The CIAO [19] یک پیاده سازی از مشخصات Light Weight CCM [7] است که برای DRS مناسب می باشد. برای حمایت از راهبردهای مختلف خدمات توصیف شده در بخش 4 و برای میسر نمودن پیکربندی انعطاف پذیر ترکیبات مناسب از آن راهبردها برای انواع برنامه ها, ما کنترل پذیرش, تنظیم دوباره حالت بیکار و خدمات توازن بار در CIAO را به صورت اجزای قابل پیکربندی پیاده سازی نموده ایم. هر مولفه, یک خدمت مشخص را با صفات قابل پیکربندی و واسطه های به طور مشخص تعریف شده برای همکاری با دیگر مولفه ها فراهم می کند و می تواند چندین بار با صفات یکسان و متفاوت نمونه برداری شود. نمونه های مولفه می توانند با هم در زمان اجرا از طریق پورت های مناسب برای تشکیل یک DRS متصل شوند.

همانطور که شکل 3 نشان می دهد, ما شش مولفه قابل پیکربندی را برای حمایت از وظایف سر هم پیوسته متناوب و نامتناوب زمان-واقعی توزیع شده با استفاده از نسخه 5.6/1.6/0.6 از ACE/TAO/CIAO طراحی و پیاده سازی نموده ایم. خط عمودی تکه تکه در شکل 3, تقسیم بندی منطقی مولفه های مدیریت وظیفه و پردازش برنامه به فرآیندهای جداگانه را نشان می دهد. در پیاده سازی ما, مدیر وظیفه می تواند روی یک پردازنده برنامه یا روی یک پردازنده جداگانه کار کند, همانطور که در شکل 1 در بخش 3 نشان داده شده است. برای کارایی, تعاملات محلی توسط فراخوان های روش پیاده سازی می شوند, در حالیکه برای انعطاف پذیری, تعاملات راه دور توسط هدایت رویداد به هم پیوسته پیاده سازی می شوند.



شکل 3. پیاده سازی مولفه

مولفه TE, وظایف ورودی را حفظ می کند, برای تصمیم مولفه AC منتظر می ماند و وظایف را آزاد می کند. مولفه کنترل پذیرش (AC) تصمیم می گیرد که آیا وظایف را بپذیرد یا خیر. مولفه توازن بار (LB), در مورد تخصیصات وظیفه برای توازن کاربردهای مصنوعی پردازنده ها تصمیم گیری می کند. مولفه First/Intermediate (F/I) Subtask, اولین وظیفه فرعی میانی را در یک اولویت معین اجرا می کند. مولفه آخرین وظیفه فرعی, آخرین وظیفه فرعی را در یک اولویت معین اجرا می کند. مولفه تنظیم دوباره بیکار (IR), کارهای فرعی کامل شده را در زمانی که یک پردازنده به حالت بیکار می رود, گزارش می کند.

هر مولفه می تواند چندین صفت قابل پیکربندی داشته باشد, به طوری که با یک بحرانیت متفاوت و زمان اجرای متفاوت نمونه برداری شود (برای مولفه های برنامه) یا یک راهبرد مختلف (برای مولفه های AC, IR, و LB). همانطور که در بخش 3 بحث نمودیم, رویکردهای کنترل پذیرش و توازن بار ما, یک معماری متمرکز را اتخاذ می کند یک نمونه مولفه AC و یک نمونه مولفه LB در حال اجرا روی یک پردازنده مرکزی را به کار می گیرد (که پردازنده مدیر وظیفه نامیده می شود).

هر پردازنده برنامه شامل یک نمونه از یک مولفه TE و نمونه از یک مولفه IR می شود. مولفه TE در هر پردازنده, ورود وظایف در آن پردازنده را به مولفه AC گزارش می دهد و سپس وظایف را بر اساس خط مشی کنترل پذیرش,

آزاد یا مردود می کند. هر وظیفه سر هم پیوسته, توسط یک زنجیره از مولفه های وظیفه فرعی F/I و یک مولفه وظیفه فرعی آخرین پیاده سازی می شود. ما در حال حاضر, رفتار هر مولفه را با جزئیات توصیف می کنیم.

مولفه TE. زمانی که یک وظیفه می رسد, مولفه TE آن را در صف انتظار می گذارد و یک رخداد "ورود وظیفه" را به رویداد "پذیرش" از مولفه AC می دهد. وظیفه متناظر در حال انتظار در صف, فوراً آزاد خواهد شود. مولفه TE دارای دو صفت پیکربندی است. یکی ID پردازنده است که نمونه های به کار گرفته شده مولفه TE در پردازنده های مختلف را متمایز می کند. دیگری, صفت در هرکار/ در هر وظیفه است که نشاندهنده اینست که آیا قبل از آزاد شدن هر کار در یک وظیفه متناوب, مولفه TE, آن را تا زمان دریافت رویداد "پذیرش" از مولفه AC نگه خواهد داشت یا خیر. اگر این صفت در حالت در هر وظیفه تنظیم شود, زمانی که یک وظیفه متناوب, پذیرفته می شود, تمام کارهای بعدی از آن وظیفه متناوب می توانند فوراً آزاد شوند. این صفات می توانند در ایجاد یک نمونه مولفه TE تنظیم شوند و نیز می توانند در زمان اجرا اصلاح شوند.

مولفه های اولین/میانی (F/I) و آخرین وظیفه فرعی.

هر دوی مولفه های (F/I) و آخرین وظیفه فرعی, وظایف فرعی را اجرا می کنند. تنها تفاوت بین این دو نوع مولفه, اینست که مولفه وظیفه فرعی F/I دارای یک پورت اضافیست که رویدادهای "تحریک" ررا برای راه اندازی اجرای وظیفه فرعی بعدی منتشر می کند. مولفه آخرین وظیفه به این پورت نیاز ندارد, زیرا آخرین وظیفه فرعی دارای یک وظیفه فرعی بعدی نیست. هر نمونه از این انواع مولفه ها شامل یک قسمت اعزام می شود که یک وظیفه فرعی خاص را در یک اولویت مشخص شده اجرا می کند. هر دو نوع مولفه دارای سه صفت قابل پیکربندی هستند. دو صفت اول, زمان اجرای وظیفه و سطح اولویت هستند که به طور نرمال در ایجاد نمونه های مولفه مشخص شده توسط توسعه دهندگان برنامه تنظیم می شوند. سومین صفت, بدون IR, IR در هر وظیفه یا IR در هر شغل است که به معنی قاعده تنظیم دوباره است که در هر وظیفه یا کار فعال نمی شود یا می شود. در هر وظیفه به معنای اینست که مولفه تنظیم دوباره حالت بیکار در زمانی که کارهای فرعی متناوب کامل شوند, اعلان نخواهد شد. چون هز کار از یک وظیفه نامتناوب به عنوان یک وظیفه غیرمتناوب مستقل با یک آزادسازی در نظر گرفته می شود, مولفه تنظیم

دوباره حالت بیکار زمانی اعلام می شود که کارهای فرعی نامتناوب کامل شوند. هر دوی مولفه های وظیفه فرعی F/I و آخرین وظیفه فرعی, روش کامل نمونه مولفه محلی IR در زمان نیاز نامیده می شوند. قسمت های اعزام در یک وظیفه فرعی F/I یا یک مولفه آخرین وظیفه فرعی توسط یک فراخوان روش آزادسازی از نمونه مولفه TE محلی یا یک رویداد تحریک از یک نمونه مولفه وظیفه فرعی F/I تحریک می شوند.

مولفه تنظیم دوباره حالت بیکار (IR). فراخوان های روش کامل را از F/I محلی یا مولفه های آخرین وظیفه فرعی دریافت می کند و رویدادها تنظیم دوباره حالت بیکار را به مولفه AC هل می دهد. دارای یک صفت, ID پردازنده است که نمونه های مولفه واقع در پردازنده های مختلف را متمایز می کند.

مولفه کنترل پذیرش (AC). رویدادهای ورود وظیفه را از مولفه های TE و رویدادهای تنظیم دوباره حالت بیکار را از مولفه های IR مصرف می کند. رویدادهای پذیرش را به مولفه های TE منتشر می کند تا آزادسازی وظیفه میسر شود. فراخوانی های روش موقعیت در مولفه LB را برای درخواست برنامه های انتساب وظیفه پیشنهادی می سازد. مولفه AC دارای یک صفت بدون LB/ LB در هر وظیفه/ LB در هر کار است که نشان می دهد آیا توازن بار فعال شده است یا خیر و اگر فعال شده است, ایا در هر وظیفه است یا در هر کار. در صورتی که صفت برای LB در هر وظیفه تنظیم شود, زمانی که یک وظیفه متناوب پذیرفته می شود انتساب وظیفه فرعی آن تعیین می شود و برای همه کارهای زیر حفظ می شود. با این حال، وظایف نامتناوب این محدودیت را ندارند زیرا آنها تنها در زمان ورود کار تک خود اختصاص داده می شود. یک مقدار LB در هر کار به معنای اینست که طرح انتساب وظیفه فرعی می تواند برای هر کار از یک وظیفه متناوب پذیرفته شده تغییر یابد.

مولفه توازن بار (LB). این مولفه, فراخوانی های روش "محل " را از مولفه AC را دریافت می کند که طرح های انتساب برای انجام وظایف خاص را بازخوانی می نماید. جزء LB برای متعادل نمودن استفاده مصنوعی در میان تمام پردازنده ها تلاش می کند و ممکن است یک طرح تخصیص قبلی را تغییر دهید زمانی که یک کار جدید از یک وظیفه می رسد. این یک برنامه انتساب را می دهد که قابل قبول است و برای به حداقل رساندن تفاوت میان بهره

وری مصنوعی در تمام پردازنده ها پس از پذیرش آن وظیفه تلاش می کند. به طریقی دیگر، مولفه LB می تواند به جزء AC بگوید که این سیستم در صورتی غیرقابل زمانبندی خواهد بود که وظیفه پذیرفته شده باشد.

6 استقرار و پیکربندی

در حالی که مولفه های قابل تنظیم ما نشان دهنده یک گام مهم به سوی خدمات میان افزار انعطاف پذیر برای مدیریت رویدادهای نامتناوب و متناوب است، توسعه دهندگان DRS هنوز هم با چالش های انتخاب بهترین ترکیب از راهبردها و مونتاژ و استقرار اجزاء روبرو هستند. بنابراین، ما یک ابزار را که انتخاب، استقرار، و پیکربندی از این مولفه ها را به طور خودکار انجام می دهد, توسعه داده ایم. ابزار ما دارای دو مزیت است: 1) اجازه می دهد تا توسعه دهندگان نرم افزار, ویژگی های DRS را مشخص کنند و به طور خودکار نقشه آنها را برای استراتژی میان افزار مناسب طرح نمایند، و 2) شناسایی ترکیبات نادرست از استراتژی خدمات را برای جلوگیری از تنظیمات نادرست میان افزار میسر می سازد. تحقق خصوصیات استقرار و پیکربندی سبک وزن CIAO, 7] OMG] موتور استقرار و پیکربندی (Dance) نامیده می شود [9]. Dance می تواند خصوصیات مونتاژ مبتنی بر XML را در اجرای استقرار و پیکربندی اقدامات مورد نیاز توسط یک برنامه ترجمه کند. مشخصات مونتاژ به عنوان توصیف کننده های کدگذاری می شوند که چگونگی ساخت DRS با استفاده از پیاده سازی مولفه های موجود کد گذاری را توصیف می کنند. اطلاعات موجود در توصیف کننده ها شامل تعداد پردازنده ها، جزء پیاده سازی برای استفاده، نوه و مکان نمونه برداری از مولفه ها، و نحوه اتصال نمونه های مولفه در یک برنامه می شود.

موتور پیکربندی جلویی. اگر چه ابزارهایی مانند CoSMIC [20] می توانند به تولید فایل های XML کمک نمایند، این ابزارها الزامات پیکربندی خدمات جدیدی که ما ایجاد کرده ایم را در نظر نمی گیرند. در نتیجه، ما، یک موتور پیکربندی خاص (نشان داده شده در شکل 4) را ارائه می دهیم که به عنوان یک جلویی برای Dance عمل می کند، و خدمات ما را برای توسعه دهندگان نرم افزار که نیاز به تنظیم پشتیبانی برنامه ریزی نامتناوب دارند پیکربندی می کند. این فرمت به Dance کمک می کند تا پیچیدگی های مرتبط با استقرار و پیکربندی خدمات ما را پیکربندی نماید. توسعه دهنده نرم افزار در ابتدا یک فایل مشخصات حجم کار را ارائه می کند که هر وظیفه پایان

به پایان و اینکه که در چه کارهای فرعی اجرا می شوند را توصیف می کند. موتور پیکربندی جلویی ما از توسعه دهنده نرم افزار می خواهد تا ویژگی های DRS را از طریق یک رابط متنی ساده که در شکل 5 نشان داده شده است مشخص کند.

موتور پیکربندی جلویی, مشخصات فایل حجم کار را تجزیه می کند و به طور خودکار نقشه ویژگی های نرم افزار مشخص شده توسط توسعه دهنده را برای تنظیمات مناسب پیکربندی برای کنترل پذیرش، تنظیم دوباره آماده به کار، و خدمات تعادل بار آماده می کند. در نهایت، طرح استقرار مبتنی بر XML تولید می شود، که می تواند توسط Dance شناخته شود. به عنوان مثال، شکل 4 یک مجموعه ای از پاسخ به این چهار سوال را نشان می دهد. بر اساس این پاسخ، خدمات AC، IR، و LB همه باید با از هر کار (PT) استراتژی پیکربندی شده استفاده نماید. شکل. 4 نیز بخشی از فایل XML تولید شده توسط موتور پیکربندی ما را با تنظیم استراتژی LB از PT، که با توجه به پاسخ های توسعه دهنده به سوالات دوم و سوم است نشان می دهد.

برای به اجرا درآوردن مهلت زمانبندی یکنواخت پایان به پایان، مولفه های وظیفه فرعی F/I و آخرین وظیفه فرعی هر دو در معرض یک ویژگی به نام "اولویت" قرار می گیرند. وقتی که موتور پیکربندی ما, مشخصات فایل حجم کار را می خواند، اولویت ها را در مرتبه مهلت های پایان به پایان به وظایف اختصاص می دهد، و این اطلاعات اولویت را برای طرح استقرار XML تولیدشده می نویسد تا بعداً توسط DAnCE تجزیه شود. موتور پیکربندی جلویی ما نه تنها مشخصات مونتاژ به خوبی شکل گرفته را با توجه به توسعه دهندگان نرم افزار و دستورالعمل ها تولید می کند، بلکه بررسی امکان سنجی در تنظیمات پیکربندی را برای اطمینان از هدایت درست محدودیت های وابسته نیز انجام می دهد. به عنوان مثال، کنترل پذیرش در هر وظیفه با هر کار تنظیم دوباره بیکار, متناقض خواهد بود همانطور که ما در بخش 4.5 ذکر نموده ایم. از آنجا که یک توسعه دهنده می تواند ترکیبات پیکربندی سرویس ناسازگار را مشخص نماید، رویکرد ما باید قادر به تشخیص و عدم مجاز نمودن آنها باشد. اگر ویژگی های نرم افزار ها توسط توسعه دهندگان ارائه نشده باشد، موتور پیکربندی ما نیز می تواند تنظیمات پیکربندی پیش فرض عرضه نماید، به عنوان مثال، در هر وظیفه کنترل پذیرش، تنظیم دوباره آماده به کار و تعادل بار.

ما از ویژگی <DAnCE configproperty> برای گسترش مجموعه ای از ویژگی ها استفاده کرده ایم که می تواند به طور انعطاف پذیر با توجه به دیگر تنظیمات پیکربندی پیکربندی شود. برای مثال، اگر خدمات موازنه بار با استفاده از راهبرد در هر وظیفه پیکربندی شود, ویژگی مربوطه از مولفه AC نیز باید در هر وظیفه تنظیم شود. DAnCE's Plan Launcher طرح استقرار مبتنی بر XML را تجزیه می کند و نام ویژگی (LB_Strategy) و ارزش در یک ساختار داده (ویژگی) را ذخیره می کند که یک حوزه از تعریف به عنوان مثال ساختار AC است. تعاریف از مثال AC و دیگر موارد جزء شامل یک طرح استقرار (استقرار :: DeploymentPlan) می شود که پس از آن برای DAnCE's Execution Manager تصویب می رسد. Execution Manager, ساختار داده های برنامه به کارگیری را به اعزام به DAnCE's node Execution Manager منتشر می کند که آن را به یک ساختار داده اولیه تجزیه می کند (NodeImplementationInfo). در نهایت، برنامه مدیریت گره, ساختار داده های مقداردهی اولیه برای برنامه گره را تصویب می کند. وقتی برنامه گره, نصب نمونه جزء AC را انجام دهد، مقدار دهی اولیه ویژگی LB_Strategy از مولفه AC را از طریق یک واسطه تنظیم استاندارد (set_configuration)، با استفاده از ساختار داده های مقدار دهی اولیه که دریافت کرده است انجام می دهد.

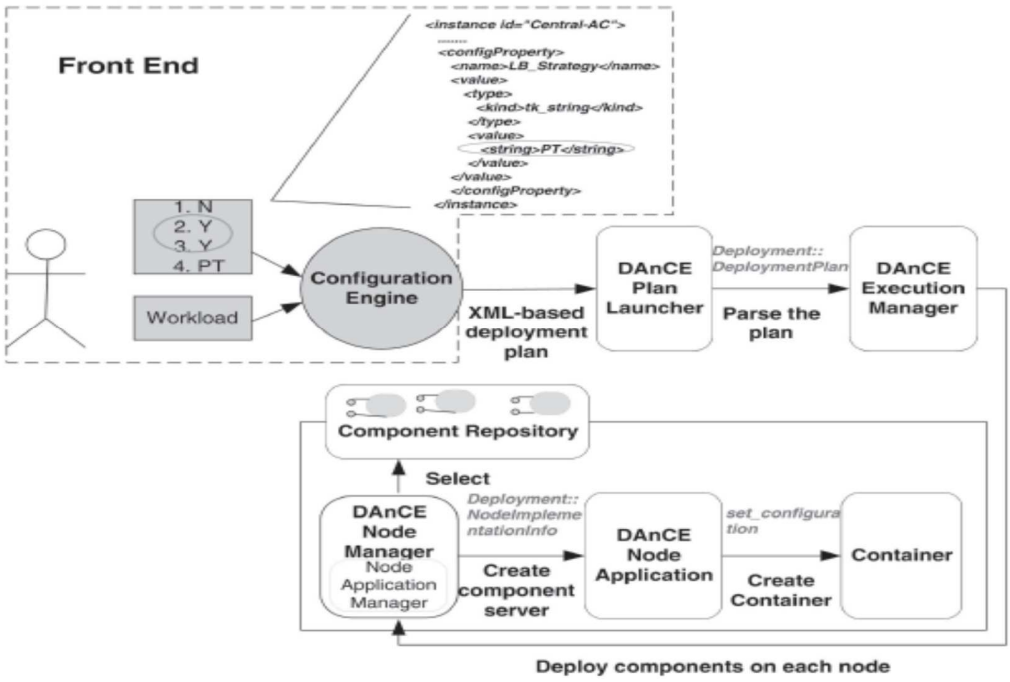
7 ارزیابی های تجربی

برای اعتباردهی رویکرد ما و برای ارزیابی عملکرد، سربار و مزایای ناشی از آن، ما یک سری آزمایشات را انجام دادیم که آنها را در این بخش توصیف می کنیم. آزمایشات بر روی یک بستر آزمایشی متشکل از شش دستگاه متصل شده توسط 100 مگابیت در ثانیه سوئیچ اترنت انجام شد. دو پنتیوم IV ماشین 2.5 گیگاهرتز با G RAM1 و 512000 کش هر دو پنتیوم IV ماشین 2.8 گیگاهرتز با RAM 1G و هر 512000 کش، و دو نفر پردازنده پنتیوم 3.4 گیگاهرتز -IV با G RAM2 و 2480000کش استفاده شدند. هر دستگاه نسخه 2.4.22 از سیستم عامل KURTLinuxرا اجرا می کند. یک پنتیوم IV 2.5 گیگاهرتز به عنوان یک مدیر وظیفه مرکزی استفاده می شود که در آن مولفه های AC و LB مستقر شده اند. پنج دستگاه دیگر به عنوان پردازنده های برنامه استفاده می شوند که TE، وظیفه F/I, وظیفه فرعی، آخرین وظیفه فرعی، و مولفه های IR مستقر شده اند.

7.1 حجم کار تصادفی

ابتدا به تصادفی 10 مجموعه از نه وظایف، هر یک شامل چهار وظایف نامتناوب و پنج وظیفه متناوب را تولید می کنیم. تعداد وظایف فرعی در هر وظیفه به طور یکنواخت بین 1 و 5 توزیع شده است. وظیفه فرعی به صورت تصادفی به پنج پردازنده نرم افزار اختصاص داده می شود. مهلت های وظیفه به صورت تصادفی بین 250 هزارم ثانیه و 10 ثانیه انتخاب می شوند. دوره های وظیفه های متناوب با ضرب الاجل ها برابر هستند. ورود وظایف نامتناوب از توزیع پواسون پیروی می کند. استفاده ترکیبی از هر پردازنده 0.5 است، اگر تمام وظایف به طور همزمان برسند. هر وظیفه فرعی به یک پردازنده اختصاص داده می شود و دارای یک پردازنده متفاوت دوبل است که به صورت تصادفی از چهار پردازنده برنامه های دیگر انتخاب می شود.

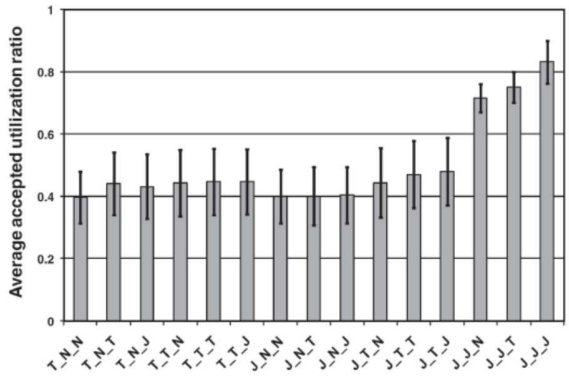
در این آزمایش، ما تمام 15 ترکیب منطقی از استراتژی را ارزیابی می کنیم, زیرا انتخاب و اجرای ترکیبات مختلف با کمک موتور پیکربندی ما راحت است. ما 10 مجموعه وظیفه را با استفاده از هر ترکیب و مقایسه آنها اجرا می کنیم. هر مجموعه وظیفه برای 5 دقیقه در هر ترکیب اجرا شد. عملکرد متریک مورد استفاده ما در این ارزیابی, نسبت استفاده پذیرفته شده است، به عنوان مثال، استفاده کل کارهای واقعاً عرضه شده تقسیم بر استفاده کل از همه کارها پس از رسیدن. به طور مختصر، از حروف بزرگ برای نشان دادن استراتژی ها استفاده می کنیم: N زمانی که یک سرویس در این پیکربندی فعال نیست. T زمانی که یک سرویس برای هر وظیفه فعال است. و J زمانی که یک سرویس برای هر وظیفه فعال می شود؛ در اشکال زیر, یک قسمت عنصر سه تایی, هر ترکیب از تنظیمات برای سه خدمت قابل پیکربندی را نشان می دهد: اول برای خدمت کنترل پذیرش, و سپس برای خدمت تنظیم دوباره حالت بیکار و آخری برای خدمت توازن بار



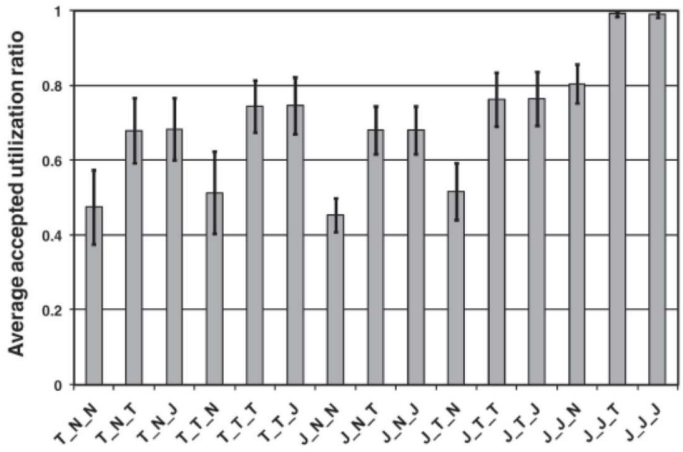
شکل. 4. موتور جلویی و تعامل آن با DANCE.

```
(1) Does your application allow job skipping?
    [yes (Y), no (N)]
(2) Does your application have replicated components?
    [yes (Y), no (N)]
(3) Does your application require state persistence?
    [yes (Y), no (N)]
(4) How much extra overhead can you accept   as it
    potentially improves schedulability?
    [none (N), some per task (PT), some per job (PJ)]
```

شکل. 5. سوالاتی برای تعیین ویژگی های انتخاب استراتژی



شکل 6. نسبت مطلوبیت پذیرفته شده برای ترکیبات مختلف (LB و AC, IR,)



شکل 7. مقایسه راهبرد LB برای ترکیبات مختلف (LB , AC, IR,)

میله ها در شکل 6 نشان دهنده نتایج متوسط روی 10 مجموعه وظیفه است. همانطور که در شکل 6نشان داده شده است. فعالسازی یا تنظیم دوباره حالت بیکار یا متعادل کردن بار می تواند مطلوبیت وظایف پذیرفته شده را افزایش دهد. علاوه بر این، این آزمایش نشان می دهد که فعالسازی IR-در هر کار $(*\_J\_*)$ به طور قابل توجهی بهتر از تنظیماتی عمل می کند که IR-در-هر وظیفه $(*\_T\_*)$ را فعال می سازند یا هرگز نه $(*\_N\_*)$. این بدان دلیل است که IR-در هر کار, سهم همه کارهای فرعی متناوب تکمیل شده را برای بهره وری مصنوعی حذف می کند که تا حد زیادی به پذیرش کارهای بیشتر کمک می کند. فعال کردن هر سه سرویس در هر کار (J J J) به طور قابل قیاس با دیگر تنظیمات (J J؟) (به طور متوسط بالاتر, هر چند اختلافات معنادار نبود) عملکرد بهتری نسبت به سایر تنظیمات داشت، حتی اگر پیکربندی J J J, بیشترین سربار را معرفی کند. ما همچنین متوجه شدیم که هنگامی که ما فقط تنظیمات مولفه LB را تغییر می دهیم و پیکربندی دو سرویس دیگر تغییر نمی کند, این تفاوت کوچک است. دلیلش این است که وقتی ما به طور تصادفی این 10 مجموعه وظیفه را تولید می کنیم، استفاده مصنوعی حاصل برای هر یک از پردازنده ها, مشابه بود. برای بررسی سود بالقوه مولفه LB، ما یک آزمایش دیگر طراحی نمودیم که در بخش بعدی توصیف می شود.

7.2 حجم کاری نامتوازن

در آزمایش دوم، ما از یک حجم کار نامتوازن استفاده نمودیم که نماینده یک DRS پویا است که در آن یک زیر مجموعه از پردازنده های سیستم ممکن است بار سنگین را تجربه کنند. مثلاً، در یک سیستم کنترل صنعتی، انسداد در شیر جریان سیال ممکن است افزایش شدید بار را بر روی پردازنده های متصل به آن، با راه اندازیه هشدار نامتناوب و وظایف تشخیصی ایجاد نماید. در این آزمایش، ما پنج پردازنده برنامه را به دو گروه تقسیم نمودیم. یک گروه شامل سه پردازنده میزبان تمام وظایف می شود. گروه دیگر شامل دو پردازنده میزبان همه وظایف تکراری می شود. ده مجموعه وظیفه به طور تصادفی در آزمایش بالا تولید می شوند، با این تفاوت که تمام وظایف فرعی به طور تصادفی در سه پردازنده نرم افزار در گروه اول قرار گرفتند و تعدادی از وظایف فرعی در هر وظیفه به صورت یکنواخت بین 1 و 3. توزیع می شود. استفاده مصنوعی برای هر یک از این سه پردازنده 0.7 است.

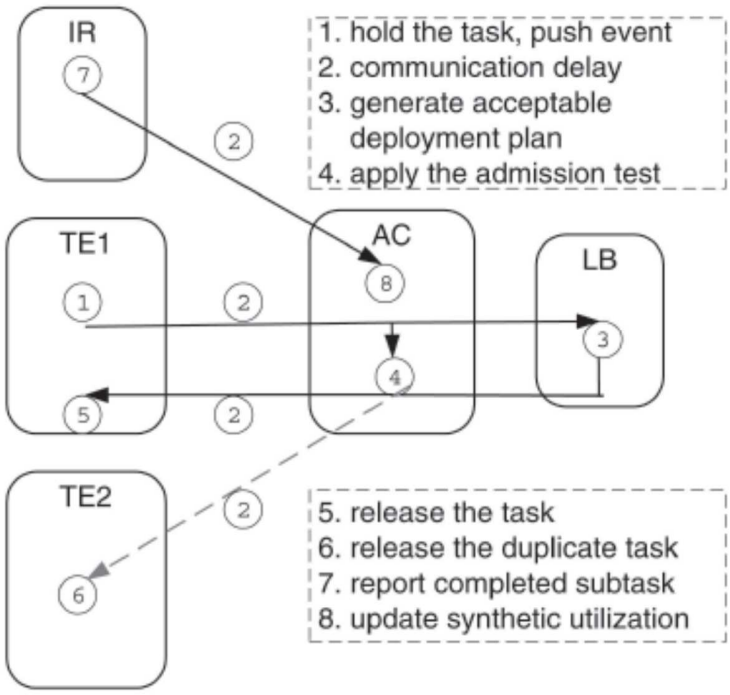
هر یک از 10 مجموعه وظیفه برای 15 ترکیب تنظیم استراتژی مختلف معتبر اجرا شد، و برای هر ترکیب ما نسبت استفاده پذیرفته شده روی 10 نتیجه گیری را متوسط گیری نمودیم. ما راهبرد موازنه بار را از هیچ به هر وظیفه، و سپس به هر کار، برای هر یک از پنج استراتژی ترکیب معتبر از کنترل پذیرش و تنظیم دوباره حالت بیکار تغییر دادیم. همانطور که شکل 7 نشان می دهد، تحت این شرایط, آزمایش مورد مطالعه ما برای LB-در-وظیفه, بهبود قابل توجهی را در مقایسه با نتایج بدون LB فراهم می کند. با این حال، تفاوت زیادی بین LB در هر وظیفه در مقابل LB در هر کار وجود ندارد. توجه داشته باشید که پنج پردازنده برنامه وجود دارد و استفاده از کار کل 2.1 است، اگر بتوانیم وظایف را تقریبا به طور مساوی میان پردازنده ها از طریق موازنه بار تمام وظایف قابل زمانبندی اختصاص دهیم (همانطور که در شکل 7 با بهره وری پذیرفته نزدیک 1.0 برای J J T و J J J نشان داده شده است). با این حال، دو پردازنده در گروه دوم, در زمانی که تعادل بار غیر فعال است، استفاده نمی شوندکه منجر به نسبت استفاده کمتر پذیرفته شده با J J N می شود.

از دو آزمایش شرح داده شده در این بخش و در بخش 7.1، ما متوجه شدیم که پیکربندی استراتژی های مختلف با توجه به ویژگی های نرم افزار می توانید یک تاثیر قابل توجه بر عملکرد یک DRS با رویدادهای نامتناوب و متناوب داشته باشد. طراحی ما از خدمات AC، IR، و LB به عنوان اجزای به راحتی قابل تنظیم اجازه می دهد تا توسعه

دهندگان نرم افزار به بررسی و انتخاب پیکربندی های معتبر بر اساس ویژگی ها و نیازهای برنامه های کاربردی، و بر اساس موازنه نشان داده شده با این نتایج تجربی بپردازند.

7.3 سربارهای مولفه های سرویس

برای ارزیابی بهره وری خدمات میان افزار مبتنی بر مولفه خود، سربارها را با استفاده از سه پردازنده برای اجرای اجزای نرم افزار و دیگر پردازنده برای اجرای مولفه های AC و LB اندازه گیری می نماییم. حجم کار به طور تصادفی با همان روش بخش 7.1 تولید می شود، جز این که تعداد کارهای فرعی در هر کار به طور یکنواخت بین 1 و 3 توزیع شده است. هر آزمایش به مدت 5 دقیقه اجرای شد. ما منابع مختلف سربار را که ممکن است در زمان رسیدن در جزء TE TE1 رخ دهد, را مورد بررسی قرار دادیم و پس از آن اجزای AC و LB, وظیفه را در TE1 جزء اجرا می کنند یا تخصیص مجدد آن را به یکی دیگر از مولفه های TE ، TE2 صورت می دهند شکل. 8 نشان می دهد که چگونه تاخیر کل برای هر خدمات شامل هزینه های عملیات واقع در چند مولفه می شود. جدول 2 تعداد عملیات نشان داده شده در شکل. 8 را برای ارائه یک حساب دقیق از تاخیر ناشی از ترکیب های مختلف از پیکربندی های خدمات نشان می دهد.



شکل 8. منبع سربار/ تاخیر

جدول 2سربارهای خدمات (میکروثانیه)

	mean	max
AC without LB (1+2+4+2+5)	1114	1248
AC with LB (1+2+3+2+5) (no re-allocation)	1116	1253
AC with LB (1+2+3+2+6) (re-allocation)	1201	1327
LB (no re-allocation) (1+2+3+2+5)	1113	1250
LB (re-allocation) (1+2+3+2+6)	1198	1319
IR (on AC side) (8)	17	18
IR (other part) (7+2)	662	683
Communication Delay (2)	322	361

برای محاسبه تاخیرات برای AC بدون LB، AC با LB بدون تخصیص مجدد، و LB بدون تخصیص مجدد، می توانیم به سادگی فاصله بین زمانی که یک وظیفه به یک پردازنده می رسد و زمانی که وظیفه بر روی پردازنده آزاد می شود را محاسبه نماییم. با این حال اگر جزء LB دوباره اولین وظیفه فرعی را در پردازنده ای مختلف با استفاده از کپی آن تخصیص دهد, همانند مورد AC با LB، تعیین یک بازه زمانی دقیق بین زمانی که یک وظیفه به یک پردازنده می رسد و زمانی که در پردازنده دیگر آزاد می شود دشوار است، به دلیل اینکه محیط آزمایش ما, هماهنگ سازی زمان با وضوح به اندازه کافی بالا را در میان پردازنده ها ارائه نمی می دهد که این محدودیت ذاتی برای بسیاری از DRS

است. در نتیجه ما سربار در همه پردازنده های درگیر را به صورت جداگانه اندازه گیری می کنیم و سپس آنها را با هم به علاوه دو بار تاخیر ارتباطات (مرحله 2 در شکل 8) بین پردازنده ها اضافه می کنیم. سه پردازنده در میان هستند: پردازنده که در آن وظیفه می رسد (مرحله 1)، پردازنده مدیر وظیفه اصلی، (مرحله 3)، و پردازنده ای که در آن وظیفه تکراری آزاد می شود (مرحله 6). ما این آزمایش را با استفاده از نسخه کورت-لینوکس 2.4.22 انجام می دهیم که شمارنده استامپ زمان حمایت شده CPU را با وضوح نانو ثانیه فراهم می کند. با استفاده از ابزار دقیق ارائه شده با توزیع کورت-لینوکس، ما می توانیم یک حسابداری دقیق از زمان شروع و توقف عملیات و تاخیر ارتباطی را به دست آوریم. برای اندازه گیری تاخیر ارتباطی بین پردازنده نرم افزار و پردازنده کنترل پذیرش در پلت فرم تجربی خود، ما یک رویداد را بین بین پردازنده نرم افزار و پردازنده کنترل پذیرش 1000 بار جلو و عقب می کنیم، پس از آن میانگین و مقدار حداکثر را در میان 1000 نتایج محاسبه می کنیم. سپس زمان رفت و برگشت 2 را برای به دست آوردن میانگین تقریبی و تاخیر ارتباطی حداکثر بین نرم افزار پردازنده و پردازنده کنترل پذیرش تقسیم می کنیم.

تاخیر کل برای LB در زمانی که تخصیص مجدد اتفاق می افتد، با همان روش مورد AC با LB با تخصیص مجدد اندازه گیری می شود. برای محاسبه تاخیر جزء IR ، اجرای آن را به دو بخش تقسیم می کنیم. سربار کوچک روی جزء کنترل پذیرش باید در تاخیر کلی محاسبه شود. سربار بزرگ در پردازنده برنامه و تاخیر ارتباطی تنهادر طول زمان آماده به کار پردازنده اتفاق می افتد، و با وجود اینکه نشان دهنده یک سربار اضافی ناشی از مولفه های IR است، بر عملکرد تاثیر نمی گذارد، و به همین دلیل ما دو قسمت را به طور جداگانه در جدول 2 گزارش نموده ایم. از نتایج در جدول 2، ما می توانیم ببینیم که همه تاخیر ناشی از خدمات قابل پیکربندی ما کمتر از 2 میلی ثانیه است که برای بسیاری از DRS قابل قبول است. برای برنامه های کاربردی با برنامه های زمانبندی کوتاه مدت، توسعه دهنده می تواند تصمیم گیری های بیشتری را در مورد نحوه پیکربندی سرویس های مبتنی بر در این اطلاعات تاخیر و بر اساس اثرات پیکربندی های مختلف در مدیریت کار صورت دهد که ما در بخش 4 در مورد آن بحث نمودیم.

8 کار مرتبط

در این بخش , ما کار مرتبط در مورد میان افزار طراحی شدهب رای مدیریت شرایط Qos برنامه ها را در نظر می گیریم که شرایط زمان واقعی آن, یک زیرمجموعه هستند. ابتدا رویکردهایی را توصیف می کنیم که بر اساس میان افزار مولفه نیستند و سپس رویکردهای مبتنی بر مولفه را در نظر می گیریم.

میان افزار آگاه از QoS. اشیای کیفیت [22] ,[21] (QuO), یک چارچوب میان افزار انطباقی توسعه یافته توسط فناوری BBN است که به توسعه دهندگان اجازه می دهد تا از تکنیک های توسعه نرم افزاری جنبه-گرا برای جداسازی نگرانی های برنامه نویسی QoS از منطق کاربرد استفاده نمایند. یک Qosket, واحد در Quo است. Quo بر تدارکات دینامیک Qos تاکید می کند در حالیکه رویکرد ما بر تدارکات QoS استاتیک تاکید می کند.پروژه dynamicTAO [23], تکنیک های انعکاسی را برای پیکربندی دوباره خط مشی های ORB و مکانیزم ها در زمان اجرا به کار می برد. مشابه با dynamicTAO, پروژه [24] Open ORB با هدف سیستم عامل های میان افزار قابل پیکربندی دوباره دینامیکی و قابل پیکربندی عمده برای حمایت از برنامه ها با شرایط دینامیک صورت می گیرد. Zhnag و Jacobson [25] نیز از تکنیک های جنبه-گرا برای بهبود سفارشی سازی مادون-ساختار هسته میان افزار در سطح ORB استفاده نمودند.

میان افزار مولفه آگاه از QoS. معماری های میان افزار مولفه برای فعال کردن فرابرنامه نویسی ویژگی های QoS با تعدادی از راه ها قوی تر شده اند. برای به عنوان مثال، تکنیک های جنبه گرا را می توان برای وصل نمودن رفتارهای مختلف [26] به ظروفی مورد استفاده قرار داد که میزبان اجزا هستند. این روش شبیه به روش ما است که در آن مکانیزم هایی برای پیکربندی ویژگی های سیستم در سطح میان افزار فراهم شده است. de Miguel [27] ظروف فعال شده-QoS را با گسترش یک واسطه ظرف EJB بیشتر توسعه داده است تا تبادل QoS مربوط به اطلاعات در میان موارد مولفه را میسر سازد. برای استفاده از این ظرف -QoS، یک جزء باید واسطه های QoSBean و QoSNegotiation را پیاده سازی نماید. با این حال، این شرط وابستگی در میان پیاده سازی های مولفه را را افزایش می دهد. پروژه اشیاء توزیع شده فعال-QoS (Qedo) [28] تلاشی دیگر برای پشتیبانی از QoS به صورت

بخشی جدایی ناپذیر از CCM است گسترش Qedo به واسطه ظرف CCM و چارچوب اجرای مولفه (CIF) نیاز به پیاده سازی مولفه برای تعامل با واسطه های ظرف QoS و مذاکره بر سر سطح قرارداد های QoS به طور مستقیم دارد. با اینکه این رویکرد برای برنامه های کاربردی خاص مناسب است، زوج های QoS را به طور محکم جفت می کند و رفتارها را برای اجرای جزء انطباق می دهد که ممکن است قابلیت استفاده مجدد از مولفه ها را محدود کند. در مقایسه، رویکرد ما به صراحت از این تزویج اجتناب می کند و ویژگی های تعریفی را در نظر می گیرد. چند تلاش های دیگر برای معرفی ویژگی های QoS در سیستم عامل های میان افزار جزء معمولی وجود داشته است. The FIRST Scheduling Framework (29] (FSF] نوشتن برنامه های مختلف و به زمانبندی برنامه منابع موجود انعطاف پذیری را پیشنهاد می دهد و در عین حالی که شرایط زمان واقعی سخت را تضمین می کند. مدل مولفه نوع زمان واقعی [30]، که امکانات QoS را در جزء ظروف ادغام می کند, نیز بر اساس مشخصات EJB و RMI معرفی شده است. الگوریتم تجزیه و تحلیل قابلیت زمانبندی 31]] برای سیستم های زمان بندی سلسله مراتبی در مولفه های وابسته که از طریق راه دور ارتباط برقرار می کنند معرفی شده است. هیچ کدام از این روش ها, خدمات قابل تنظیم برای وظایف انتها به انتهای ممتناوب و نامتناوب ترکیبی ارائه شده در رویکرد ما را فراهم نمی کند.

9 نتایج

کار ارائه شده در این مقاله نشان دهنده یک گام امیدوار به سوی خدمات میان افزار قابل تنظیم برای برنامه های کاربردی متنوع DRS با حوادث نامتناوب و متناوب است. ما یک مجموعه مشترک از ویژگی های کلیدی را به نمایندگی از بسیاری از برنامه های کاربردی DRS شناسایی کرده اند و نشان داده ایم که چگونه آن ویژگی ها به استراتژی های مناسب برای خدمات مدیریت وظیفه میان افزار زمان واقعی نگاشت می شوند. ما اجزای میان افزار قابل پیکربندی را طراحی و اجرا نموده ایم که کنترل موثر پذیرش آنلاین و توازن بار را ارائه می دهد و می تواند به راحتی بر روی سیستم عامل محاسبات توزیع شده پیکربندی و مستقر شود. موتور پیکربندی جلویی ما می تواند به طور خودکار روند مشخص ویژگی های نرم افزار را برای تولید یک طرح استقرار مربوط برای Dance توسعه دهد، لذا ساخت آن را برای توسعه دهندگان در انتخاب پیکربندی های مناسب، و جلوگیری از موارد نامعتبر آسان تر می

سازد. نتایج آزمایشات ما برای ارزیابی رویکرد ما نشان می دهد که 1) رویکرد قابل پیکربندی میان افزار مولفه ما به خوبی برای حمایت از برنامه های مختلف با ویژگیها و الزامات جایگزین مناسب است، و 2) تاخیرات تحمیل شده توسط خدمات میان افزار مولفه ما زیر 2 میلی ثانیه در می پلت فرم لینوکس است.

هدف از این پژوهش، نشان دادن اثر انواع ترکیبات استراتژی برای پیکربندی ها، با حمایت از برنامه های کاربردی با شاخص های مختلف بود. در حالی که مطالعات خاص-برنامه قطعاً بینش بیشتری نسبت به توازن بین استراتژی پیکربندی های در هر دامنه کاربرد ارائه می دهد، مجموعه وظیفه تصادفی مورد استفاده در این مقاله نشان دهنده منافع بالقوه داشتن چنین انعطاف پذیری است. نتایج ارائه شده در بخش 7 تحقیقات بیشتر برای آینده در مورد چگونگی تنظیم مجموعه کار خاص در حوزه های کاربردی در زمان واقعی مانند انتقال تصویر زمان واقعی [32]، محاسبات دریایی [11]، و ماموریت محاسبات اویونیک [33] را تشویق و تقویت می نماید.