مسئله غذا خوردن فیلسوف‌ها

تهیه و تنظیم: مبین خیبری

شماره دانشجویی: 994421017

استاد راهنما: دکتر لیلا شریفی

چکیده:

در علوم کامپیوتر مسئله فیلسوفان پشت میز غذاخوری یک مسئله تمثیلی است مربوط به طراحی هم روندی الگوریتم ها، که معمولاً برای نشان دادن مشکلات و تکنیک های همگام سازی و روش حل آن ها استفاده می شود. این مسئله در ابتدا در سال ۱۹۶۵ توسط آقای دیکسترا به عنوان یک تمرین امتحانی دانش آموزی طراحی شد و در رابطه با کامپیوترهایی ارائه شد که برای دسترسی به ابزار های جانبی درایو نواری رقابت می‌کردند.

بیان مسئله

پنج فیلسوف ساکت در اطراف یک میز قرار می گیرند. روی میز کاسه های ماکارونی وجود دارد. چنگال هایی مابین هر جفت از فیلسوف های کنار هم قرار داده شده است. هر فیلسوف باید به صورت متناوب فکر کند و بخورد. با این حال، یک فیلسوف فقط زمانی می تواند ماکارونی بخورد که که هر دو چنگال سمت چپ و سمت راست را در اختیار داشته باشد. هر چنگال در هر لحظه فقط می تواند توسط یک فیلسوف استفاده شود و بنابراین، یک فیلسوف فقط زمانی می تواند از چنگال استفاده کند که چنگال توسط فیلسوف دیگر در حال استفاده نباشد. بعد از این که یک فیلسوف خوردنش تمام شد، باید هر دو چنگال را روی میز بگذارد تا بقیه از آن ها استفاده کنند. یک فیلسوف فقط می تواند چنگال سمت راست خود یا چنگال سمت چپ خود را، زمانی که موجود باشد، در اختیار بگیرد و نمی‌تواند قبل از در اختیار گرفتن هر دو چنگال خوردن را شروع کند. مقدار خوردن ارتباطی به حجم باقیمانده ماکارونی یا فضای معده افراد ندارد؛ به عبارتی، فرض بر این است که مقدار ماکارونی نامحدود است و مقدار خوردن نیز نامحدود است. مسئله این است که چگونه یک نظم رفتاری (الگوریتم همروندی) طراحی کنیم، به گونه‌ای که هیچ فیلسوفی گرسنه نماند؛ یعنی هر کدام بتواند به مدت نامتناهی و متناوباً بخورد و فکر کند. البته با فرض اینکه هیچ فیلسوفی نمی‌داند که چه زمانی سایر فیلسوفان قصد خوردن یا فکر کردن دارند.

مشکلات

این مسئله با این هدف طراحی شد که چالش های پیشگیری از بن بست را نشان دهد. بن بست یک وضعیتی از سیستم است که در آن هیچ پیشرفتی امکان پذیر نیست .برای اینکه ببینیم راه حل این مسئله چندان آسان نیست، فرض بگیرید که هر فیلسوف باید مطابق با دستورالعمل زیر رفتار کند:

تا زمانی که چنگال چپ موجود نشده، فکر کن. سپس، زمانی که چنگال چپ موجود شد، آن را بردار؛

تا زمانی که چنگال راست موجود نشده است، فکر کن. و زمانی که موجود شد آن را بردار؛

زمانی که هر دو چنگال را به دست آوردی، مدت زمان مشخصی بخور؛

سپس چنگال راست را روی میز بگذار؛

و سپس چنگال چپ را روی میز بگذار؛

دوباره از ابتدا شروع کن.

این راه حل وسوسه انگیز با شکست مواجه خواهد شد، زیرا به سیستم این امکان را می دهد که در وضعیت بن‌بست قرار بگیرد که در آن هیچ پیشرفتی امکان پذیر نیست. این وضعیت زمانی رخ می دهد که هر فیلسوف چنگال سمت چپ را برداشته است و منتظر است تا چنگال سمت راست موجود شود. با دستورالعمل های داده شده در بالا، احتمال ایجاد این وضعیت وجود دارد و زمانی که این وضعیت رخ دهد، هر فیلسوف باید منتظر فیلسوف دیگر در سمت راست بماند تا یک چنگال آزاد شود.

اگر یک فیلسوف خاص به دلیل مشکل زمانبندی قادر نباشد تا هر دو چنگال را به دست آورد ممکن است قحطی منبع بصورت مستقل از بن‌بست نیز رخ دهد .برای مثال ممکن است یک قانون وجود داشته باشد که فیلسوف ها بعد از انتظار به مدت ۱۰ دقیقه برای موجود شدن چنگال دیگر، چنگالی را که در اختیار دارند، روی میز بگذارند و ۱۰ دقیقه دیگر صبر کنند و آنگاه مجدداً تلاش کنند. این پروتکل باعث از بین رفتن احتمال بن بست می شود، زیرا سیستم همیشه می تواند به یک وضعیت متفاوت تغییر کند. اما هنوز مشکل قفل زنده وجود دارد. اگر هر ۵ فیلسوف دقیقاً در یک لحظه وارد اتاق غذاخوری شوند و در یک لحظه هر کدام از آنها چنگال سمت چپ خود را بردارد، آنگاه فیلسوف ها ۱۰ دقیقه صبر میکنند و سپس تمام آنها همزمان چنگال های خود را روی میز می گذارند و سپس ۱۰ دقیقه دیگر صبر می کنند و مجدداً همه آنها چنگال ها را برمی دارند.

انحصار متقابل ایده اساسی مسئله است. فیلسوف های پشت میز غذاخوری در واقع یک سناریوی عمومی و انتزاعی کاربردی برای توصیف مشکلاتی از این دست را فراهم می کنند. مشکلاتی که این فیلسوف ها با آن مواجه می شوند، مشابه مشکلاتی است که در برنامه نویسی واقعی کامپیوتر (زمانی که چندین برنامه نیاز به دسترسی انحصاری به منابع مشترک دارند) رخ می دهد. این مشکلات در برنامه نویسی همروند مورد بحث قرار می‌گیرند. این مسئله در ابتدا مربوط به وسایل خارجی نظیر درایو های نواری بود. با این حال، مشکلاتی که به وسیله مسئله ی فیلسوفان پشت میز غذاخوری شبیه سازی می شوند، در

زمانی که چندین پروسه به داده هایی که در حال بروزرسانی هستند دسترسی پیدا می‌کنند، بسیار بیشتر رخ می دهد. سیستم های پیچیده نظیر کرنل های سیستم عامل از هزاران قفل و همگام سازی استفاده می کنند که نیازمند پیروی دقیق از روشها و پروتکل ها است تا مشکلاتی نظیر بن‌بست، قحطی، و خراب شدن داده به وجود نیاید .

راه حل منبع سلسله مراتبی

این راه حل، همان راه حلی است که در ابتدا توسط دیکسترا پیشنهاد شد. در این راه حل یک ترتیب سهمی به منابع (در این مثال چنگال ها) اختصاص داده می شود و این قانون را اعمال می کند که تمام منابع باید به ترتیب درخواست شوند و اینکه هیچ دو منبع نامرتبط از نظر ترتیب، هیچ وقت توسط یک واحد کار منفرد، به طور همزمان استفاده نشوند. در اینجا منابع (چنگال ها) از ۱ تا ۵ شماره گذاری می شوند و هر واحد کار (فیلسوف) همیشه از بین دو چنگالی که قرار است استفاده کند، اول چنگال با عدد کوچکتر و سپس چنگال با عدد بزرگتر را بر می دارد. ترتیب قرار دادن چنگال ها روی میز توسط هر فیلسوف اهمیتی ندارد. در این مورد اگر ۴ نفر از ۵ نفر به طور همزمان، چنگالی با عدد کوچکتر را بردارند، فقط چنگالی که بالاترین عدد را دارد روی میز باقی می ماند. بنابراین، چنگالی نصیب فیلسوف پنجم نخواهد شد. علاوه بر این، فقط یک فیلسوف به چنگالی که بالاترین عدد را دارد دسترسی دارد و بنابراین قادر خواهد بود تا با استفاده از دو چنگال غذا بخورد .

اگرچه راه حل منبع سلسله مراتبی، مانع از بن بست می‌شود اما این راه حل همیشه عملی نیست، مثلاً زمانی که لیست منابع مورد نیاز از قبل کاملاً مشخص نیست. برای مثال، اگر یک واحد کار، منابع ۳ و ۵ را در اختیار بگیرد و سپس مشخص شود که نیاز به منبع ۲ دارد، آنگاه باید قبل از به دست آوردن منبع ۲، منبع ۵ و سپس۳ را آزاد کند و سپس باید منابع ۳ و ۵ را مجدداً به همان ترتیب به دست آورد. اگر برنامه های کامپیوتر را که به تعداد زیادی از رکورد های پایگاه داده دسترسی دارند ملزم کنیم تا قبل از دسترسی به یک رکورد جدید تمام رکوردهای دارای عدد بالاتر را رها کنند، آنگاه برنامه به خوبی کار نمی‌کند، و بنابراین این روش برای این هدف غیرعملی است .

راه حل منبع سلسله مراتبی، منصفانه نیست. اگر فیلسوف ۱ در برداشتن چنگال کند باشد، و اگر فیلسوف ۲ در فکر کردن و برداشتن چنگال ها سریع باشد، آنگاه فیلسوف ۱ هیچگاه قادر نخواهد بود تا هر دو چنگال را بدست آورد .راه‌حل منصفانه باید این تضمین را بدهد که هر فیلسوف بتواند تا ابد بخورد ، حتی اگر سرعت آن در مقایسه با بقیه کندتر باشد .

راه حل حکمیت

روش دیگر این است که شرایطی ایجاد کنیم که یک فیلسوف در هر لحظه یا هر دو چنگال رو بردارد یا هیچکدام را بر ندارد. برای این روش نیاز به یک حاکم مثلاً یک گارسون داریم. یک فیلسوف برای برداشتن چنگال ها باید از گارسون اجازه بگیرد. تا زمانیکه فیلسوف ها هر دو چنگال مورد نیاز خود را بردارند، گارسون در هر لحظه فقط برای یک فیلسوف اجازه صادر می‌کند. فیلسوفان همیشه اجازه دارند تا چنگال را روی میز بگذارند. گارسون را می توان با استفاده از یک mutex پیاده سازی کرد. این روش علاوه بر نیاز

به یک ماهیت مرکزی جدید (گارسون)، می تواند منجر به کاهش موازی گرایی نیز شود: این کاهش، زمانی رخ می دهد که یک فیلسوف در حال خوردن باشد و یکی از فیلسوف های مجاور آن درخواست چنگال کند، در این حالت، تمام فیلسوفان دیگر باید منتظر بمانند تا این درخواست برآورده شود، حتی اگر چنگال برای آنها مهیا باشد.

راه حل Chaney وMisra

در سال ۱۹۸۴ Chandy و Misra یک راه حل متفاوت را برای مسئله فیلسوفان پشت میز غذاخوری پیشنهاد کردند. در این راه حل، برخلاف راه‌حل دیکسترا، به عوامل دلخواه (P_1, ..., P_n) اجازه داده می‌شود تا برای تعداد دلخواهی از منابع رقابت کنند. مضافاً اینکه، این راه‌حل کاملاً توزیع شده است و نیاز به هیچ گونه مدیریت مرکزی بعد از شروع شدن ندارد. با این حال، در اینجا برخلاف قبل، فیلسوف ها می توانند با یکدیگر ارتباط برقرار کنند (به دلیل پیام های درخواست) .

۱ .برای هر جفت از فیلسوف ها که بر سر یک منبع رقابت می کنند، یک چنگال درست کنید و آن را به فیلسوفی بدهید که n) ID برای عامل P_n) پایین‌تری دارد. هرچنگال می تواند تمیز یا کثیف باشد. در ابتدا تمام چنگال ها کثیف هستند .

۲ .زمانیکه یک فیلسوف می خواهد از مجموعه ای از منابع استفاده کند (یعنی بخورد)، باید چنگال ها را از رقیبان همسایه خود بگیرد. برای تمام چنین چنگال هایی که فیلسوف مورد نظر ندارد باید یک پیغام درخواست بفرستد .

۳ .زمانیکه یک فیلسوف که دارای یک چنگال است، یک پیغام درخواست دریافت می کند، اگر چنگالش تمیز باشد آن را نگه می دارد و اگر کثیف باشد آن را می دهد. اگر این فیلسوف بخواهد چنگال را بدهد، باید ابتدا آن را تمیز کند .

۴ .بعد از این که خوردن یک فیلسوف تمام شد، تمام چنگال های آنها کثیف می شود. اگر فیلسوف دیگری پیش از این در خواست یکی از این چنگال ها را کرده باشد، فیلسوفی که خوردن اش تمام شده است، چنگال را تمیز می کند و به وی می دهد .

از این رو، این راه حل درجه بالایی از همه روندی را فراهم می کند و یک مسئله که به اندازه دلخواه بزرگ باشد، را حل می کند .

همچنین مشکل قحطی را نیز برطرف می‌کند. برچسب تمیزی یا کثیفی به عنوان شیوه ای برای دادن ارجحیت به گرسنه ترین پروسه عمل می کند و برای پروسه هایی که تازه خوردن شان تمام شده است، خوب نیست. می توان هر کدام از این روش ها را با روشی مقایسه کرد که در آن، به فیلسوف ها اجازه داده نمی‌شود تا پشت سر هم دوبار بخورند و در عین حال در این بین به دیگران اجازه استفاده از چنگال ها را ندهند. راه‌حل Chandy و Misra در مقایسه با آن قابل انعطاف تر است، اما دارای یک عنصر است که به آن سو تمایل دارد .

آنها در آنالیز خود، یک سیستم سطوح ارجحیت را از توزیع چنگال ها و وضعیت تمیزی یا کثیفی آنها طراحی کردند. آن ها نشان دادند که این سیستم ممکن است یک گراف بدون حلقه جهت دار را نشان دهد و اگر چنین باشد، عملیات موجود در پروتکل آن ها نمی‌تواند گراف را تبدیل به یک گراف حلقوی کند. این وضعیت قطعاً موجب بن‌بست نخواهد شد. با این حال، اگر این سیستم در یک وضعیت کاملاً قرینه شروع شود، نظیر آنچه که در آن فیلسوف ها چنگال های سمت چپ خود را نگه داشته بودند، آنگاه گراف در شروع، حلقوی است و راه حل آنها مانع از بن‌بست نمی‌شود. شروع کردن سیستم به‌گونه‌ای که فیلسوف هایی که دارای شماره شناسایی پایین تری هستند، صاحب چنگال های کثیف باشند، موجب می‌شود تا گراف در ابتدا غیر حلقوی باشد .

پایان.