یک الگوریتم تحمل پذیر دربرابر خطا جهت ایجاد انحصارمتقابل توزیع شده

منصوراسماعیل پور
عضو هیات علمی دانشگاه آزاد اسلامی همدان
ma_esmaeilpour@hotmail.com

محمدرضا میبدی
عضو هیات علمی دانشگاه صنعتی امیرکبیر
maybodi@ce.aut.ac.ir

چکیده

الگوریتمهای زیادی جهت حل مشکل انحصار متقابل در سیستم توزیع شده ارائه شده است اما در آنها تعداد پیامهای ارسالی خیلی بالا بوده و پیچیدگی زمانی بالایی دارند، در این مقاله ما الگوریتمی نوین جهت حل این مشکل ارائه می کنیم که تعداد پیامهای ارسالی برای بدست آوردن ناحیه بحرانی، از درجه لگاریتمی می باشد. این الگوریتم تحمل پذیر دربرابر خطا بوده و با خراب شدن فرآیندها می تواند دوباره بازسازی گردد و به کار خود ادامه دهد. در نهایت نشان خواهیم داد که این الگوریتم آزاد از بن بست و قحطی زدگی می باشد.

واژه های کلیدی : الگوریتم توزیع شده ، قفل گذاری ، درخت شبهه کامل

۱. مقدمه

داشتن مسئله انحصار متقابل در سیستم های توزیع شده امری ضروری است و به دلیل عدم وجود حافظه مشترک در اینگونه سیستم ها تنهاراه، جهت ارتباط بین گره ها ارسال پیام می باشد. در سیستمهای توزیع شده، الگوریتمهای زیادی جهت گرفتن ناحیه بحرانی وجود دارند که به سه دسته زیر تقسیم می شوند:

الف) الگوریتم متمرکز : یکی از گره ها(هر فرآیند در سیستم یک گره نامیده می شود) نقش هماهنگ کننده را داشته و دیگر گره ها جهت کسب انحصار متقابل باید از وی اجازه بگیرند.

ب) الگوریتم توزیع شده[10] : باید بنحوی از تمام گره ها برای رفتن به ناحیه بحرانی اجازه گرفت یا بتوان گره های دیگر را قفل کرد. الگوریتمی که ما در این مقاله ارائه می کنیم در این رده قرار می گیرد.

ج) الگوریتهای مبتنی بر توکن[11] : از یک حلقه منطقی بین گره ها استفاده می کند که یک توکن نوبت نها را تعویض می کند Tanenboum[8].

۲. مروری بر کارهای گذشته

الگوریتمی که به وسیلهRicart&Agrawala[2] پیشنهاد شد جهت بدست آوردن ناحیه بحرانی از۲(n-1)پیام استفاده می کرد کهn تعداد فرایندهابود. (n-1) پیام جهت درخواست و(n-1) پیام پاسخ درخواست. در ضمن این روش مبتنی بر قانون رای گیری عمومی بود و هر گره جهت کسب ناحیه بحرانی می بایست با تمام گره ها ارتباط بر قرار کند. در الگوریتم دیگری که به وسیلهGiford[3] Skeen[4] مطرح شد به جای گرفتن اجازه از گره های با اولویت، گرفتن بیشترین رای از گره های دیگر را مورد استفاده قرار داد هر گره که بیشترین را به دست آورده باشد می تواند وارد ناحیه بحرانی شود[5].

Lampor[1] از برهه زمانی منطقی جهت پیاده سازی انحصار متقابل استفاده کرد. که این پروتکل از قحطی زدگی در به دست آوردن منابع جلوگیری می کند. Raymond[6] الگوریتمی را معرفی کرد که از درخت پوشای شبکه برای پیاده ساری توکن استفاده کرده بود و نشان داد که متوسط تعداد پیامهای ارسالی از درجه o(logn) بود(دربهترین شرایط) و در شرایط بدتر، برابر قطر شبکه اغزایش پیدا می کرد.

الگوریتم معرفی شده توسطMeakawa[7] نیز حالتی از رای گیری است که به $\sqrt{N}$پیام جهت کسب ناحیه بحرانی نیازدارد.

۳. نحوه کار مدل

۳-۱. ایجاد گروه و کسب ناحیه بحرانی

هر فرآیند که ایجاد شد، شماره ای یکه به آن نسبت داده می شود Lamport[۱]. هر گره ایی که بعدازایجاد شدن خراب شود اگر دوباره وارد سیستم گردد باید عددی دیگر به آن نسبت داده شود یعنی همانند گرههای جدید با او برخورد می گردد. به فرض که گره ایی با شماره i ایجاد شود، هرگره بعداز آنکه یک عدد یکه به آن نسبت داده شد، گروهی برای خود تشکیل می دهند. نحوه ایجاد گروه بصورت برنامه زیر است.

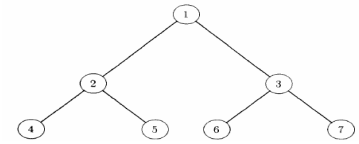
```
While  i >1  do
  Begin
     i:=  [ i / 2 ]
     sendmsg_rq(i)
  End
```

برنامه I(ارسال پیام به هر گره)

این برنامه درهر گره بعداز ایجاد شدن انجام می شود. در نهایت برای ساخت یک گروه۲-L۲پیام ارسال می‌گردد که L سطح قرارگیری گره i در درخت است که چون درخت حالتی شبهه کامل است پس حداکثر سطح درخت همان عمق درخت یا برابر۱+$[\log_2 n]$. الگوریتم حتی طوری طراحی شده است که نیازی به ایجاد گروه نداشته باشد و بدون ایجاد گروه نیز کار خود را ادامه می دهد.

اولین موردی که در گرفتن ناحیه بحرانی باید مورد توجه قرار گیرد مسئله انحصار متقابل است و اینکه این الگوریتم تضمین کند که انحصار

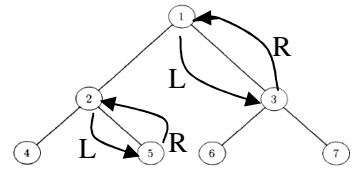
متقابل برقرار است و هیچ دو فرآیندی نمی‌توانند همزمان وارد ناحیه بحرانی شوند. فرض که هر گره ای برای خود گروهی تشکیل دهند. طبق برنامه I (شکل منطقی قرارگیری گره‌ها بصورت درخت زیر است)



(شکل ۱: گروهبندی گره های درخت)

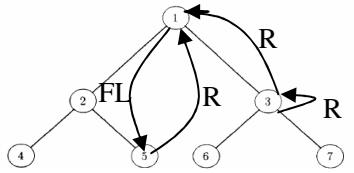
G1={1} , G2={2,1} , G3={3,1}
G4={4,2,1},G5={5,2,1} G6={6,3,1} G7={7,3,1}

فرض که همزمان گره ۵ و ۳ قصد بدست آوردن ناحیه بحرانی را داشته باشند. هر کدام از گره‌ها باید اعضای گروه خودشان را قفل نمایند تا بتوانند ورد ناحیه بحرانی گردند. گره ۵ ابتدا خودش را قفل کرده و سپس اقدام به قفل گره ۲ می کند، از طرف دیگر گره ۳ خودش را قفل کرده و گره یک که هم گروهش است را نیز قفل می‌نماید و گره۳ چون تمام اعضای گروه و خودش را بدرستی قفل کرده می‌تواند وارد ناحیه بحرانی گردد (شکل ۲).



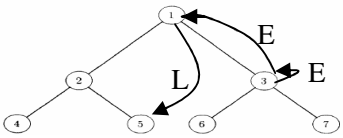
(شکل ۲: درخواست قفل)

از آن طرف ۵ اقدام به قفل هم گروه دیگرش یعنی گره ۱ می‌کند.



(شکل ۳: ارسال جواب قفل)

گره ۱ خود توسط گره‌ایی دیگر قفل شده است، درنتیجه جواب عدم قفل را به گره ۵ ارسال می‌کند و درخواست۵ را به صف می‌برد(شکل۳).



(شکل ۴: آزادسازی قفل)

پس این تضمینی کرد که همیشه فقط یک فرآیند در ناحیه بحرانی به سر می‌برد. حال اگر کار گره ۳ تمام شود و از ناحیه بحرانی خارج گردد پیام آزادسازی به گره‌های قفل شده توسط خود را ارسال می‌کند(شکل ۴). گره ۱ بعد از دریافت پیام آزادسازی از گره قفل کننده به صف خود نگاه می‌کند وگره ۵ را از صف برمی‌دارد و پیام "آیا هنوز قصد وارد شدن به ناحیه بحرانی را داری" را به آن ارسال نموده و اگر دوباره قصد وارد شدن به ناحیه بحرانی را داشته باشد پیام قفل را به گروه ۱ ارسال کرده و تنها عضو باقیمانده گروهش را قفل می‌کند و وارد ناحیه بحرانی می‌گردد.

۳-۲. بررسی بن بست، گرسنگی وانتظار محدود

فرض که گره ۳ و گره ۲ همزمان درخواست قفل گره یک را نمایند، گره یک هر دو درخواست را گرفته و درخواستی که از فرآیندی با شماره کوچکتر می‌باشد را در اولویت قرار می‌دهد در نهایت گره ۲ موفق به قفل گره یک شده و درخواست گره ۳ به صف می‌رود و هرگاه گره ۲ از ناحیه بحرانی خارج شود گره یک پیام آزادسازی از گره ۲ را دریافت می‌کند و گره ۳ را از صف خارج کرده و گره ۱ برای گره ۳ قفل می‌شود، گره ۳ وارد ناحیه بحرانی می‌گردد. پس در این الگوریتم هیچ بن بستی وجود ندارد زیرا به هر حال طبق اولویتی که وجود دارد بن بست درخواست قفل بین چندین فرآیند شکسته می‌شود و همچنین هیچ گرسنگی و انتظار نامحدودی نیز پیش نمی‌آید و هیچ گره‌ایی به اندازه نامحدود منتظر رفتن به ناحیه بحرانی نیست زیرا طبق شماره گذاریهایی که انجام شده و طبق اولویتی که یاد شده است این مسئله نیز بدرستی رفع می‌گردد.

۳-۳. بررسی شرط پیشرفت و عادلانه بودن آن

فرآیندی که از ناحیه بحرانی خارج می‌شود و در بخش پایانی خود قرار می‌گیرد باید اجازه دهد گره‌های دیگری که هنوز وارد ناحیه بحرانی نشده‌اند، وارد ناحیه بحرانی گردند و خودش دوباره تلاش نکند. این مسئله به صورت زیر قابل حل است :

برای هر فرآیند و نسبت به ناحیه بحرانی که مورد استفاده آن است می توان محدودیت هایی قایل شد و آن اینکه هر فرآیند بعد استفاده از ناحیه بحرانی شماره خود (اولویت خود) را پس داده و شماره ای دیگر را بگیرد. در این حالت همه فرآیندهای شرکت کننده در رقابت از این لحظه به بعد می توانند او را قفل نمایند. این کار تا زمانی ادامه می یابد تا کلیه فرایند های درگیر از ناحیه بحرانی استفاده کنند و کارشان تمام شود. خلل ناشی از این کار که در درخت ایجاد می گردد با همان ترفند اولیه که یاد شد برطرف می گردد.

۳-۴. تحمل پذیری در برابر خطا

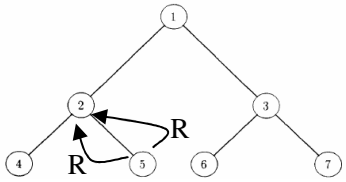
اساسی‌ترین مشکلاتی که در یک سیستم توزیع شده پیش می‌آید خراب شدن هرکدام از گره‌ها بدون اطلاع گره‌های دیگر است. درادامه نشان می‌دهیم که این الگوریتم کاملا تحمل پذیر خطا بوده و در برابر خرابی گره‌ها خللی در روند کار ایجاد نمی‌گردد.

L = Lock, FL = Free Llock, R = Request
E = rElease

خطایی که احتمال رخداد دارند بصورت زیر است.

۳-۵.گره‌هایی که خراب شود، دیگران چگونه مطلع می‌گردند

گره‌هایی که خراب شده است حداقل در یک گروه قرار دارد و اگر گره‌هایی بخواهد ناحیه بحرانی را بدست آورد و گره خراب شده در آن گروه باشد خرابی را کشف می کند (شکل ۵).



(شکل۵: بازسازی درخت)

در این شرایط تعداد پیامهای ارسالی با در نظر گرفتن تحمل پذیر خطا برابر با: حداکثر logn-1 پیام جهت قفل logn-1 پیام جهت جواب قفل ودرصورت خرابی گره ای دربدترین شرایط (گره ریشه خراب باشد)یک پیام جهت اطلاع از خرابی گره و سرانجام بعداز کشف خرابی، logn-2 پیام آزادسازی گره هایی که قفل کرده بود. پس درکل تعداد پیامها برابر :

logn-1+logn-1+1+logn-2=3logn-3=3(logn-1)

و گروهش از $G5=\{5,2,1\}$ به$G_2=\{2,1\}$ تغییر پیدا می‌کند و درصدد قفل گره یک بر می‌آید. درنهایت پس از اضافه شدن گره‌های دیگری به سیستم، خود به خود جای گره ۵ پر می شود.

این الگوریتم کاملاً تحمل پذیری در برابر خطای فرآیندهای دیگر را تضمین می‌کند و چه در بهترین شرایط و یا بدترین شرایط که شامل خرابی گره‌ها و خراب شدن فرآیندهای مختلف است خللی در الگوریتم پیشنهادی ایجاد نمی‌کند و تعداد پیامهای ارسالی را ثابت و از درجه لگاریتمی نگه می‌دارد که در ادامه به تفصیل به تعداد پیامهای ارسالی خواهیم پرداخت.

۳-۶. تفاوت سطح گره درخواست کننده با گره خراب شده بیشتر از یک سطح باشد

مثال گره ۱۱ درخواست ناحیه بحرانی را نماید و گره ۲ خراب شود. گره ۱۱ خودش و گره ۵ را قفل می‌کند و قصد قفل گره ۲ را دارد که می‌بیند گره ۲ خراب شده است. طبق الگوریتم پیشنهادی باید گره ۱۱ جایگزین گره ۲ شود. $G11=\{11,5,2,1\}$ که البته باید عدد خودش را به گره ۲ تغییر دهد و خودش یعنی ۲ را قفل کند. باید قفل گره‌های سطح زیرین یعنی ۵ را باز نماید و گروه G_{11} به $G2=\{2,1\}$ تغییر گروه داده شود و هنگام بروز رسانی گروه G_{11} به G_2 باید گره ۵ نیز از قفل درآید (پیام آزادسازی توسط گره ۱۱) و در نهایت با حذف گره ۵ از گروه G_{11} سرانجام با تغییر نام گره ۱۱ به گره ۲ همراه هستیم.

تنها مشکلی که در بازسازی گروهها بعد از خرابی و کشف یک گره وجود دارد اینست که باید گره جایگزین شده، تمام خواص گره خراب شده را بدست آورد که البته منظور از تمام خواص یعنی اینکه گره خراب شده قبل از خراب شدن، آیا توسط گره‌هایی دیگر قفل شده است یا نه. چون گره‌هایی که جایگزین گره‌هایی خراب شده می‌گردند عملاً تمامی نقش آن گره را از این به بعد باید بازی کنند، پس باید خواص او را نیز بدست آورد.

۳-۷. بعد از کشف هر خرابی جهت اطلاع از خواص گره ای که خراب شده، می‌توان از فرزندان آن گره در درخت منطقی سوال کرد

برای مثلا اگر گره i خراب شده باشد و کشف شود، جهت اطلاع حاصل کردن از خواص آن می‌توان از گره‌های$i*۲$ یا $i*۲+۱$ سوال کرد تا خواص گره i بدست آید چون ما یا از مسیر$i×۲$ یا $i×۲+۱$ آمدیم پس نیازی نیست که از گره ای که در مسیر ما بوده سوال کنیم و فقط کافی است از گره ای که در مسیر دیگری بود سوال نماییم.

۳-۸. تعداد پیامها جهت اخذ ناحیه بحرانی چقدر است

L-۱ پیام جهت درخواست قفل، L - ۱ پیام جواب قفل، اگر یک گره در این مسیر خراب شده باشد باید گروه به روز شود که در بدترین حالت L-2 پیام آزادسازی برای گره‌های مابین گره درخواست کننده و گره خواب شده مورد نیاز است. بعد از حذف گره‌های یادشده، گروه جدید تشکیل می‌شود. اگر گره خواب شده را i فرض کنیم که جدیدا کشف شده است یک پیام به$i×۲$ یا$i×۲+۱$ جهت اطلاح از وضعیت قبلی گره خراب شده ارسال می‌کنیم، یک پیام اطلاع به گرهi از$i×۲$ یا $i×۲+۱$ ارسال می‌شود و اگر پیام تایید قفل را بفرستند آنگاه ۲ پیام هم جهت اطلاع از اینکه آیا گره درخواست کننده قبلی اکنون در ناحیه بحرانی است یا خیر مورد نیاز است پس در این حالت درمجموع $(L-1)+(L-1)+(L-2)+2+2$ پیام مورد نیاز است که برابر 3L پیام می‌شود، که در بدترین شرایط L-۱ پیام آزادسازی هم بعد از خارج شدن گره از ناحیه بحرانی لازم است که در مجموع تعداد پیامها **4L-۱** است که چون عمق درخت $[\log_2 n]+1$ است پس در بدترین شرایط برابر $4\log_2 n$ خواهدشد. در اینجا یک مسئله دیگر وجود دارد و آن اینست که گره‌ایی که خراب شده است خود در ناحیه بحرانی باشد.

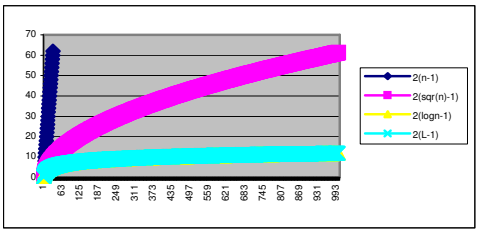
برای مثال فرض که گره ۸ طبق (شکل۱) قصد وارد شدن به ناحیه بحرانی را دارد و قبلاً گره ۴ در ناحیه بوده و خراب شده است. طبق موارد ذکر شده گره ۸ خرابی گره ۴ را کشف کرده و خود را به گره ۴

تغییر نام و تغییر گروه می‌دهد و از گره‌های سطح زیرین که دو طرف دیگر قرار دارد سوال می‌کند ولی پیام عدم قفل را دریافت می‌کند و برنامه I را انجام می‌دهد و می‌خواهد از گره ۲ جهت رفتن به ناحیه بحرانی اجازه بگیرد که می‌بیند گره ۲ در قفل گره ۴ بوده پس اطلاع پیدا می‌کند که قبلا گره ۴ در ناحیه بحرانی بوده و خراب شده است، دوباره قفل را حفظ می‌کند و جهت قفل کردن گره‌های بالا دستی توسط برنامه I ادامه می‌دهد که می‌توان از این لحظه کار قفل کردن گره‌های دیگر را انجام ندهد و از همین لحظه وارد ناحیه بحرانی شود و مطمئن است که گره‌ایی دیگر وارد ناحیه بحرانی نمی‌شود چون هر گره خواص خود را حفظ کرده، ولی برای اطمینان از کار، دوباره برنامه I را برای گره‌های بالادستی انجام می‌دهد. مطمئناً موفق خواهد شد و تمام گره‌های روی مسیر را دوباره قفل می‌کند و وارد ناحیه بحرانی می‌گردد. پس همان تعداد پیامها حفظ شده و خللی در الگوریتم ایجاد نمی کند.

۴. مقایسه الگوریتم پیشنهادی با الگوریتمهای موجود

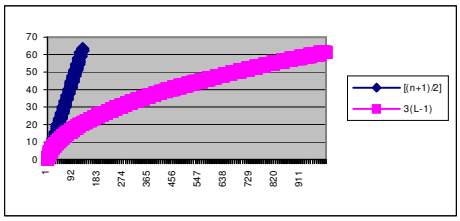
الگوریتمهایی که در این راستا ارائه شده اند زیادهستند ولی تعداد کمی از آنها پیشرفتی نسبت به دیگر الگوریتمها داشته اند. ما در این فصل الگوریتم پیشنهادی را با تعدادی از الگوریتهای ارائه شده شاخص مقایسه می کنیم.

- در بهترین شرایط الگوریتم Recart&Agrawala جهت گرفتن ناحیه بحرانی نیاز به 2(n-1) پیام دارد درحالی که الگوریتم Meakawa نیاز به 2(√n-1) و الگوریتم El-Abbadi[9] نیاز به 2(logn-1) پیام دارد و الگوریتم پیشنهادی حداکثر تعداد پیامی که نیاز دارد برابر2(L-1) که L سطح گره درخواست کننده در درخت منطقی است و حداکثر برابر2(logn-1) یا برابر همان الگوریتم El-Abbadiمی باشد، مقایسه آن در شکل ۶ آمده است.



(شکل ۶: مقایسه بهترین حالت)

- در بدترین شرایط اگر در گرفتن ناحیه بحرانی دچار خرابی گره یا گره هایی شویم الگوریتم Recart&Agrawala تحمل پذیر دربرابر خطا نیست و سیستم دیگر بدرستی کار نمی کند و انحصار متقابل و مسایل مربوطه را پشتیبانی نمی کند و همین موارد نیز برای الگوریتم Meakawa صادق است ولی در الگوریتم El-Abbadi با محاسبه سربار تحمل پذیری خطا تعداد پیامهای ارسالی برابر [(n+1)/2] می باشد و این در حالی است که تعداد پیامهای ارسالی در الگوریتم پیشنهادی با محاسبه سربار تحمل پذیر خطا برابر3(L-1) یا 3logn-3 شده است که به مراتب به صرفه تر وکم هزینه تر است، مقایسه آن درشکل۷ آمده است.



(شکل ۷: مقایسه بدترین حالت)

۵. نتیجه گیری

در این الگوریتم نشان دادیم که تعداد پیامهای ارسالی برای بدست آوردن ناحیه بحرانی، از درجه لگاریتمی بوده و الگوریتم تحمل پذیر دربرابر خطا است و با خراب شدن فرآیندها می تواند دوباره بازسازی گردد و دیدیم که الگوریتم آزاد از بن بست و قحطی زدگی بود.

۶. منابع

1.Lamport , L. Time,clocks, and ordering of events in a distributed system. Commun ACM 21. 7(July 1978),558-565.

2.Ricart, G., and Agrawala, A K An optimal algorithm for mutual exclusion in computer networks Commun. ACM 24, 1(Jan.1981), 9-17.

3.Gifford, D.K. Weighted voting for replicated data. In proceedings of the seventh ACM symposium on operating systems principles (Dec.1979), 150-159.

4.Skeen, D. Non-blocking commit protocols. In proceedings of the ACM SIGMOD conference on Management of data (June 1982),133-147.

5.Garcia-Molina, H., and Barbara, D. How to assign votes in a distributed system. J.ACM 32,4(Oct.1985),841-860.

6.Raymond, K A tree-based slgorithm for distributed mutual exclusion. ACM Trans Comput. Syst. 7.1(Feb.1989),61-77.

7.Maekawa. M A √N algorithm for mutual exclusion in decentralized system. ACM Trans. Comput. Syst. 3,2(May 1985), 145-159.

8.Tanenbaum, A S. *distributed operating systems* ,2d ed, prentice-Hall, 1995.

9.D. Agrawala and A. El-Abbadi, "An efficient and fault-tolerant solution for distributed mutual exclusion," *ACM Trans. Comp. Syst*., vol. 9, no. 1, pp. 1-20, Feb. 1991.

10.P.C.Saxena, G.Rai, A Survy of Permission-based Distributed Mutual Exclusion Algorithm,Computer Standard & Tnterface,V.25 n.2, p 159 – 181, may 2003.

11.M.Benchaiba,A.Bouabdallah, N.Badache,M.Ahmed-Nacer, Distributed Mutual Exclusion Algorithm in mobile ad hoc network. an overview, ACM SIGOPS Operating System Review, V.38 n.1, p.74 – 89, gaunary 2004.