مقدمه

امروز همه به دنبال سرعت و دقت بالا در محاسبات می باشند با وجود اینکه پیشرفتهای بسیاری در سالهای اخیر رخ داده ولی همچنان نیاز به سرعت بیشتر و پردازش های real time در حال افزایش است. بنا بر همین دلایل ابر کامپیوتر ها بوجود آمدن که کمک های بسیاری به انسان ها کرده اند ولی استفاده از ابر کامپیوتر ها در برخی موارد مشکلاتی دارند که جهت انجام این پردازش ها ما را به سوی سیستم های کلاستر می کشاند.

برخی ابزارهای برنامه نویسی پردازش موازی نیز ما را در این امر یاری می کنند که در این پروژه از کتابخانه تبادل پیام MPICH استفاده شده است.

پردازش مورد نظر این پروژه حذف نویز از تصاویر که دارای مقیاس بزرگ می باشد مانند تصاویری که بوسیله ماهواره ها تهیه می شود. در این پروژه می خواهیم بوسیله شکستن اطلاعات و تقسیم پردازش بین پردازنده ها مختلف زمان پردازش را کاسته و بر سرعت پردازش بی افزایم. بستر مورد استفاده، یک شبکه مبتنی بر TCP/IP می باشد.

جهت حذف نویز از تصاویر با بررسی فیلتر های مختلف، سه فیلتر انتخاب شده و پس از پیاده سازی مورد تجزیه و تحلیل قرار گرفته اند. در این بررسی دو زمان مد نظر قرار گرفته است که زمان تلف شده جهت ارتباطات و انتقال اطلاعات می باشد و دیگری زمان پردازش می باشد.

فصل اول

الگوریتم های پردازش تصویر

۱-۱ مقدمه

برای حذف نویز از تصاویر می توانیم از الگوریتم های فراوانی استفاده کنیم که بعضی از آنها به آسانی پیاده سازی و بعضی دیگر دارای پیاده سازی پیچیده تری هستند این الگوریتم ها بنا به کاربردهایی که دارند مورد استفاده قرار می گیرند، دارای نتایج و عملکردهای متفاوتی هستند. در این بخش سعی می کنیم چند الگوریتم متداول برای حذف نویز از تصاویر را به اجمال توضیح دهیم.

۱-۲ الگوریتم میانگین همسایه ها[۱]

در این الگوریتم اساس کار بر مبنای ساختار فایلهای تصویری و پیکسل ها می باشد و به این صورت عمل می کند که طبق عدد گرفته شده در همسایگی پیکسل مرکزی مربعی را فرض می کند که این عدد می تواند ۳،۵ ویا ۷ باشد(به طور مثال اگر عدد گرفته شده ۳ باشد مربعی ۳*۳ که ۸ پیکسل اطراف مرکزی میشود را در نظر می گیرد) و سپس مقادیر این پیکسل ها را باید با یکدیگر جمع نموده و در نهایت این مجمع را بر تعداد پیکسلهای جمع شده تقسیم می کند. در واقع اگر G(X,Y) تابع حاصل از نتایج باشد که جایگزین خانه مرکزی می شود و F(X,Y) نقاط اطراف نقطهٔ مرکزی باشند. M تعداد نقاط باشد آنگاه الگوریتم را به صورت زیر خواهیم داشت:

$$G(X,Y)= \sum_{n,m \, \pounds \, S} \quad F(n,m)$$

این الگوریتم باعث از بین رفتن نویز نمی شود بلکه باعث می شود تا نویز به طور یکنواخت در کل تصویر پخش شود.

۱-۳ الگوریتم فیلتر میانه[۲]

این الگوریتم نیز بر اساس ساختار پیکسل های فایل کار می کند و دقیقا مانند الگوریتم میانگین همسایه ها یک مربع به دور یک پیکسل مرکزی در نظر می گیرد ولی با این تفاوت که این بار عملیات میانگین گیری صورت نمی گیرد بلکه مقادیر خانه ها را در یک آرایه ریخته و آنها را مرتب می کند خانه ای که دارای مقدار میانه آرایه است را به عنوان خانه مرکزی قرار می دهد و بقیه نقاط نیز به همین تر تیب اعمال می شوند. این الگوریتم بنا به نوع الگوریتم مرتب سازی دارای زمان مصرفی $O(n^2)$ یا O(n Log(n)) می باشد در نتیجه از الگوریتم قبلی زمان بیشتری را مصرف می کند که دارای زمان

---

۱. Neighborhood Averaging
۲. Median Filter

مصرفی O(n) بود ولی در اینجا به دلیل ساختار و نوع الگوریتم و اینکه نقاط به ترتیب مرتب سازی می شوند به مرور نویز از تصاویر حذف می شوند در صورتی که درالگوریتم قبلی نویز ها از تصاویر خذف نمی شدند.

برای درک بهتر الگوریتم Median در شکل ۱-۱ یک مثال آورده شده است.

۱۲۲	۱۲۵	۱۲۷
۱۲۴	124	۱۲۵
۱۱۹	۱۲۱	۶

Median 3*3 ⟶

۱۲۲	۱۲۵	۱۲۷
۱۲۴	۲۴۰	۱۲۵
۱۱۹	۱۲۱	۶

شکل ۱-۱ عملکرد الگوریتم Median

پس از مرتب سازی عدد ۱۲۴ در میانه آرایه قرار گرفته و جایگزین خانه مرکزی می شود.

۱-۴ الگوریتم فیلتر پایین گذر[۳]

در این الگوریتم یک دایره به مرکز پیکسل مورد نظر می گیرد و بر اساس یک تابع محاسبه می کند که کدام یک از نقاط داخل دایره قرار می گیرند و کدامیک خارج ایره نقاطی که داخل دایره هستند را در نقطه مرکزی ضرب می کند و نتیجه را باز می گرداند، این الگوریتم بیشتر برای تقویت نقاط تضعیف شده تصویر کاربرد دارد کارایی که ما از آن می خواهیم را نخواهد داشت.

۱-۵ الگوریتم میانگینی از چندین تصویر[۴]

این الگوریتم زمانی به کار می رود که چندین نمونه از یک تصویر داشته باشیم و به این صورت عمل می کند که با مقایسه این تصاویر و یافتن نقاط متناظر در تصاویر آنها را به عنوان نتیجه قرار می دهد و بهترین گزینه را از بین چندین تصویر انتخاب می کند و مهمترین مشکل این الگوریتم نیز در دسترس نبودن چندین نمنه از یک تصویر می باشد.

بنا به عملکرد الگوریتم Median و نوع استفادهای که مورد نیاز ما که بیشتر مقصود حذف نویز از تصاویر و مقایسه الگوریتمهای سریال و موازی و تاثیر حجم پردازش و حجم داده در روند موازی سازی بود الگوریتم مناسبی به نظر می رسد .

---

۳ . Lowpass Filter

۴ . Averaging OF Multiple Image

۱-۶ الگوریتم نقطه میانی[۵]

این الگوریتم مانند الگوریتم Median ابتدا مقادیر پیکسل های داخل پنجره را مرتب کرده و سپس کوچکترین عنصر را با بزرگترین عنصر جمع کرده و میانگین آن دو را بر می گرداند.

۵. Midpoint

فصل دوم
برنامه نویسی موازی بر مبنای تبادل پیام

۲-۱ مقدمه:

استفاده از تکنیک ارسال پیام (Message passing) در برنامه هایی که روی ماشینهای موازی خصوصا ماشینهای دارای حافظه اشتراکی (Shared Memory) اجرا می شود، امروزه امری متداول و مرسوم می باشد، در این تکنیک ، پردازنده های مختلف یک ماشین موازی برای برقراری ارتباط با یکدیگر، از ارسال پیام استفاده می کند. با توجه به آنکه سرعت انتقال اطلاعات ما بین پردازنده ها نسبت به سرعت پردازش پردازنده ها ، بسیار پایین می باشد، ارسال پیام به عنوان یک گلوگاه حیاتی و مهم در اجرای برنامه های موازی مطرح می شود که لازم است به شکل جدی مورد توجه قرار گیرد، یکی از راههای بخورد با این مشکل روش سخت افزاری است، بدین معنی که با افزایش سرعت بسترهای ارتباطی، کاهش فاصله ما بین پردازنده ها، طراحی توپولوژی سیستم به گونه ای که ارسال پیام در آن به صورت بهینه و کارآمدی صورت پذیرد و ... به حل این معضل بپردازیم. در کار این راه حل، پیاده سازی نرم افزاری ارسال پیام نیز می تواند به عنوان راهی دیگر مطرح شود. هر اندازه که این پیاده سازی بهینه و مناسب با توپولوژی سخت افزار سیستم صورت گرفته باشد، سرعت ارسال پیام بیشتر خواهد بود. از طرف دیگر به طور کلی ارسال پیام نیازمند کد نویسی دقیق و بدون خطا می باشد و با توجه به آنکه در نوشتن یک برنامه موازی پیچیدگیهای خاص موازی سازی همواره مطرح است. درگیر شدن با جزئیات چنین پیاده سازیهای چندان خوشایند نیست، با توجه به نکات بیان شده، نیاز به کتابخانه هایی که قابلیت بر قراری ارتباط ما بین پردازنده های یک سیستم موازی را به صورت بهینه و دقیق داشته باشند ضروری به نظر می رسد. تلاشهای گوناگونی در این زمینه صورت پذیرفته است که حاصل آنها دو گزینهMPI ،PVM می باشد.

MPI یا Message Passing Instruction استانداردی است که اولین بار در سال ۱۹۹۴ و با همکاری ۴۰ شرکت مختلف بنا نهاده شد. در این استاندارد کتابخانه ای از توابع لازم به منظور پیاده سازی تکنیک ارسال پیام به صورت بهینه، عملی و دقیق تعریف شده است. با تعریف این استاندارد، شرکتهای مختلفی به پیاده سازی آن پرداختند و بدین ترتیب نسخه های متعددی ازآن به بازار عرضه شد که بسیاری از آنها به صورت رایگان در اختیار می باشد. از طرفی دیگر ،PVM یا Parallel Virtual Machine که سابقه طولانی تری نسبت به MPI دارد، برای اولین بار سال ۱۹۸۹ و در آزمایشگاه Oak Ridgeتولید شده است ، بر خلاف MPI که یک استاندارد تعریف شده است که پیاده سازیهای مختلفی دارد PVM از ابتدا به صورت یک کتابخانه پیاده سازی شده مطرح شده است، البته در طی زمان و با بروز کردن کاربردهای جدید، نسخه های متعددی از PVM نیز تولید شده اند که برای استفاده در زبان های مختلف برنامه نویسی و یا روی سخت افزارهای خاص به کار می رود.

۲-۲ معرفی MPI و استانداردهای مختلف آن

MPI یک کتابخانه تبادل پیام می باشد، و یک طرح استاندارد جهت استفاده درسیستم های توزیع شده و تبادل پیام و محاسبات موازی است. هدف از ایجاد یک رابط تبادل پیام ساده، یک استاندارد جامع به منظور نوشتن برنامه های تبادل پیام است. و موارد زیر را مد نظر قرار گرفته شده است:

۱. کاربردی بودن (عملی)
۲. قابلیت انتقال

۳. کارایی
۴. انعطاف پذیری

تاریخچه:

MPI نتیجه تلاش های افراد و گروه های بیشماری در مدت بیش از دو سال می باشد.

- از سال ۱۹۸۰ تا اوایل سال ۱۹۹۰: حافظه های توزیعه شده و محاسبات موازی توسعه می یافتند و از ابزار های ناسازگار جهت نوشتن این برنامه ها استفاده می شد. معمولا یک رقابتی بین شاخص های قابلیت انتقال،عملکرد،کارایی و هزینه وجود داشت در این حین نیاز به یک استاندارد احساس می شد.
- در آپریل سال ۱۹۹۲: در کارگاهی که موضوع بحث آن استاندارد های تبادل پیام در محیط های حافظه توزیعه شده بود و توسط مرکز تحقیقات محاسبات موازی پشتیبانی می شد. شاخص های الزامی و اساسی استاندارد رابط تبادل پیام بحث شد و یک گروه به منظور ادامه فرآیند استاندارد سازی بوجود آمد. سپس پیش نویس طرح توسع ایجاد شد.
- در نوامبر ۱۹۹۳: در کنفرانس ابرمحاسبات ۹۳ پیش نویس استاندارد MPI ارائه شد.
- آخرین نسخه از این پیش نویس در می سال ۱۹۹۴ انتشار یافت و از طریق آدرس ذیل در دسترس قرار دارد.(http://www.mcs.anl.gov/Projects/mpi/standard.html)
- دلایل استفاده از MPI:
  - استانداردسازی: MPI تنها کتابخانه تبادل پیام مطرح شده بر اساس یک استاندارد می باشد. کامپیوتر های HPC نیز به صورت مجازی آن را پشتیبانی می کنند.
  - قابلیت انتقال: در صورت انتقال از یک سیستم به سیستمی با ساختار متفاوت در صورت پشتیبانی از MPI نیازی به تصحیح کد برنامه نیست.
  - کارائی: در پیاده سازی می توان از ویژه گی های خاص سخت افزار جهت بهبود کارائی استفاده نمود.
  - جامعیت (بیش از ۱۱۵ روال)
  - دسترسی: پیاده سازی های مختلفی از آن در دسترس می باشد، در حوض دستفروشی و عمومی (تعداد بالا)
- زیربنای سیستم حافظه های توزیعه شده میباشد که شامل: بیشمار ماشین موازی، SMP cluster،workstation clusters، heterogenous networks می باشد.
- صراحت در موازی کاری ها: برنامه نویس مسئول تشخیص صحت عملیات موازی و پیاده سازی آن و نتیجه حاصل از پیاده سازی آن بوسیله MPI می باشد.
- تعداد وظایف اختصاص یافته شده به یک برنامه موازی ایستا می باشد و در هنگام اجاری برنامه امکان ایجاد وظیفه جدید نمی باشد.
- توانایی استفاده از زبان های c و Fortran را دارا می باشد.

۳-۳ معرفی پیاده سازی MPICH

MPICH اولین پیاده سازی صورت پذیرفته از استاندارد MPI می باشد. در حقیقت یکی از اهداف پیاده سازی MPICH آن بود که استاندارد مورد نظر را به سرعت به صورت عملی مورد آزمایش قرار دهند و نقاط ضعف آن را پیش از آنکه استفاده از آ« فراگیر شود. مشخص کنید تا امکان رفع سریع آن وجود داشته با شد.

از اهداف دیگر پیاده سازی MPICH، توجه به کارایی بالایی که تولید شده بوده است.

در این پیاده سازی با در نظر گرفتن محدودیتهای اعمال شده از سوی استاندارد MPI، سعی شده است که کلیه محدودیتهای سخت افزاری کنار زده شده و از سخت افزار مورد استفاده به شکل بهینه و کارآمدی استفاده شود. هدف دیگر پیاده سازی MPICH ، قابلیت انتقال (Portability) آن بوده است. در حقیقت نام این پیاده سازی (Chameleon) نیز به همین دلیل انتخاب شده است، زیرا آفتاب پرست مظهر قابلیت انطباق با محیط می باشد.

آخرین نسخه MPICH، نسخه 1.2.5.7 آن می باشد که استاندارد MPI 1.2 را به صورت کامل پشتیبانی می کند و برخی قسمتهای MPI 2 را هم پیاده سازی کرده است.

و شامل دو ورژن تحت Unix و تحت WINDOWS می باشد. از نظر امکانات یک محیط جامع برنامه نویسی موازی است. این پیاده سازی دارای ابزارهایی به منظور دنبال کردن کد (Code Tracing) و نیز تولید فایلهای ثبت وقایع(Log file) می باشد. همچنین ابزارهایی به منظور نمایش میزان کارایی کد موازی نوشته شده و تست و تصحیح آن تعبیه شده است.

از مزایای بسیار مهم MPICH آن است که کد آن به صورت رایگان در اختیار است. در حقیقت بسیاری از پیاده سازیهای دیگر MPI، کد رایگان MPICH را به عنوان مبنای برای پیاده سازی خود قرار داده اند. نسخه تحت ویندوز MPICH قابلیت کامپایل شدن و اجرا شدن در دو محیط Visual C++ ، Fortan را دارد دو نسخه تحت Unix آن، بر روی کلیه ماشین های که Unix روی آنها قابل نصب است، تحت کامپیلرهای c (gcc) و Fortan(f77) قابل کامپایل و اجرا می باشد.

فصل سوم

پیاده سازی الگوریتم

۳-۱ مقدمه

در این بخش سعی می کنیم روش های بکار برده شده و ابزارهای مورد استفاده را شرح دهیم و تا حدودی نتایج بدست آمده از الگوریتم را نمایش دهیم و در پایان با مقایسه الگوریتم موازی شده با الگوریتم سریال خواهیم دید که موازی سازی تا چه حدی می تواند موثر باشد.

۳-۲ کامپایلر و کتابخانه ها

چون الگوریتم نوشته شده باید در دو حالت سریال و موازی Compile و اجرا می شود بنابراین برای الگوریتم سریال از کامپایلر gcc (کامپایلر استاندارد C تحت سیستم عامل Linux) همراه کتابخانه های استاندارد C استفاده کرده ایم ولی برای موازی سازی الگوریتم می توانیم دو روش را به کار ببریم، یکی اینکه تمام امکانات را خودمان پیاده سازی کنیم که در این صورت باعث می شود کار ما بسیار وقت گیر، غیر استاندارد و بدون قابلیت جابجایی می باشد. دیگری استفاده از کتابخانه استاندارد MPICH و کامپایلر MPICC می باشد که تنها تفاوت MPICC با gcc در این است که از توابعی که امکانات موازی سازی را به ما می دهد پشتیبانی می کند.

۳-۳ سخت افزار

بنا بر دلایلی که در دو بخش گذشته گفته شد ما به سخت افزار خاصی نیاز نیست و با حداقل سیستمی که قابلیت اجرای سیستم عامل Linux و کتابخانه استاندارد MPICH داشته باشد و همچنین قابلیت اتصال به یک شبکه را ایجاد کند می توانیم کار خود را به انجام برسانیم. ولی جهت مقایسه بهتر نتایج از سیستم های استفاده می کنیم که دارای مشخصات سخت افزاری یکسان به شرح زیر می باشند:

fetuer of hardware's	
CPU	intel p4-full 3.2 64 bit
random memory acess(RAM)	1 giga bayte ddr2 bus 677
network interface	10/100mb fast ethernet

جدول ۳-۱ مشخصات سخت افزاری

۳-۴ فایل BMP

۳-۴-۱ دلایل استفاده از فایل BMP

ذخیره سازی تصاویر به فرمت ها و قالب های گوناگون ممکن است که اکثر این قالب ها حالت فشرده شده به تصویر می دهند و قابلیت دسترسی به مقادیر پیکسلهای تصاویر وجود ندارد. تنها فرمت تصویری که امکان دسترسی به مقادیر آن وجود دارد فرمت BMP می باشد که مقادیر را به صورت واقعی و پشت سر هم در فایل ذخیره می کند.

۳-۴-۲ ساختار فایل BMP

یک فایل BMP از سه قطعه اصلی و مهم تشکیل شده که در ذیل توضیح خواهیم داد:

- قطعه(header) سرآیند: علامت مشخص کننده فایل BMP عبارت BM6 می باشد که در اول قطعه Header آورده می شود بعد از این عبارت دو مقدار از نوع Longint که نشان دهنده طول و عرض تصویر و یک مقدار از نوع Byte که نشان دهنده Bit depth می باشد از تعداد بیتهای که به هر Pixel اختصاص داده می شود وجود دارد مقدار Bit depth می تواند سه حالت داشته باشد:
  ۱. یک است که به هر پیکسل یک بیت اختصاص داده می شود که یا سفید است یا سیاه است.
  ۲. ۸ است که نشان دهنده تصاویر 256 با Gray scale می باشد.
  ۳. ۲۴ باشد که نشان دهنده تصاویر RGB است.
- قطعه(pallet) تخت رنگ : این قطعه تنها در صورتی که مقدار Bit depth برابر ۸ باشد وجود دارد و در صورت وجود دارای 256 مقدار longint است که قابلیت تعریف ۲۵۶ رنگ را به ما می دهد.
- قطعه داده (Data): قطعه اصلی فایل می باشد که در بر گیرنده مقادیر پیکسل های می باشد. مقادیر این پیکسل ها به صورت Scan Line هایی که از پایین به بالا هستند و در داخل این Line Scan ها اطلاعات پیکسل ها از چپ به راست قرار می گیرد.

۳-۵ الگوریتم

برنامه ما باید به گونه ای با شد که به تعداد پردازنده ها وابستگی نداشته باشد و کاملا پارامتری باشد این پارامتر ها که به نام Notasks در برنامه ما آورده شده مقدار خود را به طور خود کار در هنگام اجرا از کامپایلر MPICC در یافت می کند همچنین علاوه بر این پارامتر، هر پردازنده پارامتر دیگری با نام rank خواهد داشت که نشان دهنده شماره ترتیب process آن از میان تعداد کل Process های موازی شده می باشد.
Processای که بر روی کامپیوتر Master اجرا می شود دارای rank=0 می باشد و این تنها راه تشخیص پروسه Master از Client ها می باشد.

۳-۵-۱ پروسه Master

وظیفه شکستن تصویر به تعداد پردازنده ها و ارسال هر باریکه تصویربه آنها بر اساس ID پردازنده ها و در نهایت جمع آوری نتیجه عملیات از پردازنده ها و نوشتن آنها در یک فایل می باشد. در هنگام شکستن اطلاعات به مشکلی بر می خوریم که نحوه شکستن و لبه هایی که در اثر شکستن بوجود می آید پدید آورنده این مشکل هستند در قسمتهای بعد به این مسئله خواهیم پرداخت.

۳-۵-۲ الگوریتم موازی

Start

این قسمت توسط کامپیوتر اصلی اجرا می گردد:

```
If(master process)
                {
  Open source file
  Check source file format
   If (error source or target file)
Exit
  Read from file each line of all processes & send
  Receive data from all process & write to file
```

}

این قسمت توسط تمامی کامپیوترها اجرا می گردد:

Receive each line
Run filter on pixels
Send back to process master
End

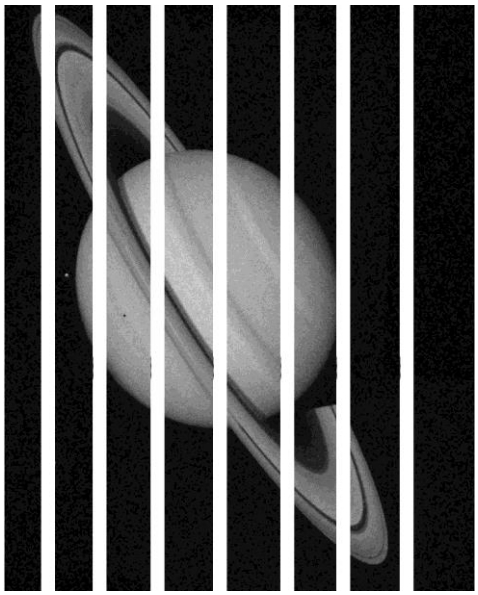
۳-۵-۴ الگوریتم سریال

Start
Open source file
Check source file format
Create target file
If (error source or target file)
Exit
Read lines
Run filter on pixels
Write to file filtered pixels
}
End

۳-۶ شکستن تصاویر

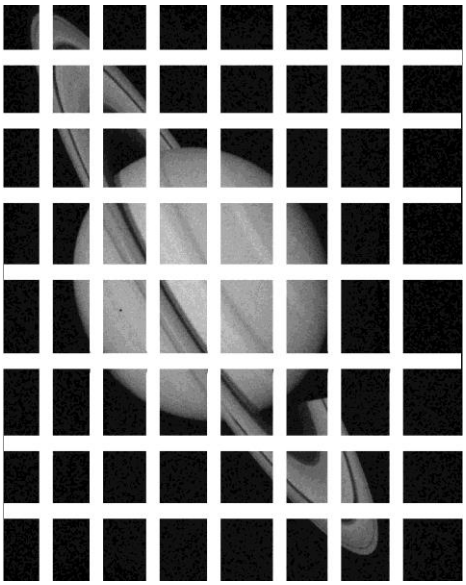
برای شکستن یک فایل تصویر بین چند پردازنده به ۳ حالت می توانیم این کار را انجام دهیم.

- شکستن تصاویر به صورت عمودی: بنا به مطالبی که در ساختار فایل BMP گفته شد امری بسیار دشوار است به این دلیل که هر Scan Line باید به قسمت های تقسیم شود و در نتیجه در لبه های بوجود آمده به مشکل پیدا کردن خانه ها مجاور برخورد خواهد کرد.



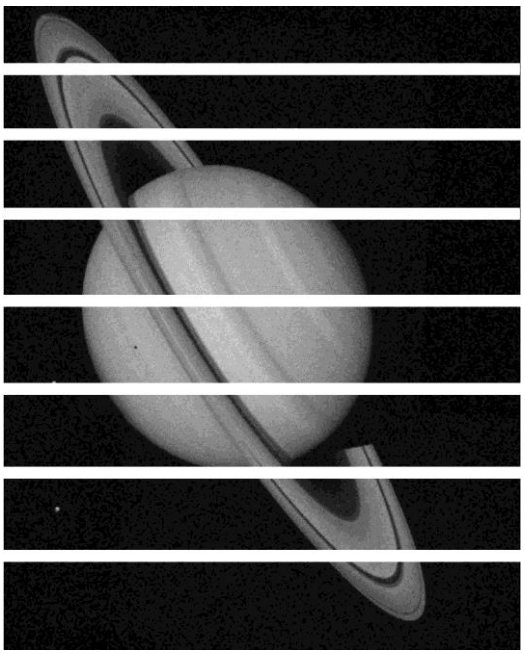
شکل ۳-۱ شکستن تصاویر به صورت عمودی

- شکستن تصاویر به صورت شطرنجی: مهمترین مشکل این حالت را می توان به وجود آمدن لبه های زیاد در حین عمل تقسیم بیان کرد. اما در حالتی که تصویر با اندازه بزرگ اشته باشیم و بخواهیم داده ها را بین پردازنده های زیادی تقسیم کنیم ناگزیر باید از این روش استفاده کنیم.



شکل ۳-۲ شکستن تصاویر به صورت شطرنجی

- شکستن تصویر به صورت افقی: این حالت کاملا مطابق با ساختار فایل BMP می باشد چرا که اطلاعات در فایل Scan Line،BMP به صورت افقی تعریف می شود در نتیجه تقسیم این فایل در حالت معمول به صورت افقی می توان بهترین گزینه باشد.



شکا ۳-۳ شکستن تصاویر به صورت افقی

بنابراین در الگوریتم پیاده سازی شده از روش شکستن تصویر به صورت افقی استفاده کرده ایم.

۳-۷ لبه ها

در الگوریتم مبتنی بر پنجره برای حذف نویز، مقدار هر پیکسل با پیکسل های مجاور سنجیده می شود بنا براین باید به پیکسل های همجوار آن دسترسی داشته باشیم در هنگام شکستن Pixel هایی که در سطر اول هر باریکه تصویر قرار می گیرد به پیکسل های موجود در Scan Line پایین خود دسترسی ندارد این امر در مورد سطر آخر هر باریکه تصویر نیز بوجود می آید این مقدار باید به نحوی برای Processها فراهم گردد که برای حل این مشکل می توان از روشهای زیر استفاده کنیم:

- هر پروسه این مقادیر را از پروسه های همجوار خود دریافت کند. برای انجام این امر باید کامپیوترهایی که پروسه ها بر روی آنها انجام می گیرد ارتباطی دو طرفه و دائم داشته باشد. در این حالت پردازشی که باید صورت بگیرد بسیار بیشتر و وقت گیر است.
- این مقادیر مستقیما از طریق برنامه Master فرستاده شود. در این حالت نیازی به ارتباط Clinetها با یکدیگر نیست. و در نتیجه ساختار شبکه سادهتر و زمان پردازش کوتاهتری خواهیم داشت.

در هر دو حالت ارسال لبه ها برای پردازنده ها زمانی را مصرف خواهد کرد که این زمان در اگوریتم ۷*۷ بیشتر و در ۵*۵ کمتر و در ۳*۳ به حداقل خود می رسد. در الگوریتم ۷*۷ به ازای هر لبه به ۶ Scan Line اضافه در ۵*۵ به ۴ Scan Line اضافه و در ۳*۳ به ۲ Scan Line اضافه نیاز داریم.

۳-۸ تحلیل نتایج

برای مقایسه اینکه موازی سازی الگوریتم چقدر در سرعت انجام عملیات تاثیر و تا چه حد موازی سازی برای الگوریتم ما مفید است. الگوریتم موازی و سریال را بر روی یک تصویر ۸۵۶ مگابایتی که دارای ۳۰۰۰۰*۳۰۰۰۰ پیکسل می باشد.

۳-۸-۱ الگوریتم های ۷*۷

زمان اجرای الگوریتم سریال حذف نویز از تصویر برای سه الگوریتم زیر برابر است با

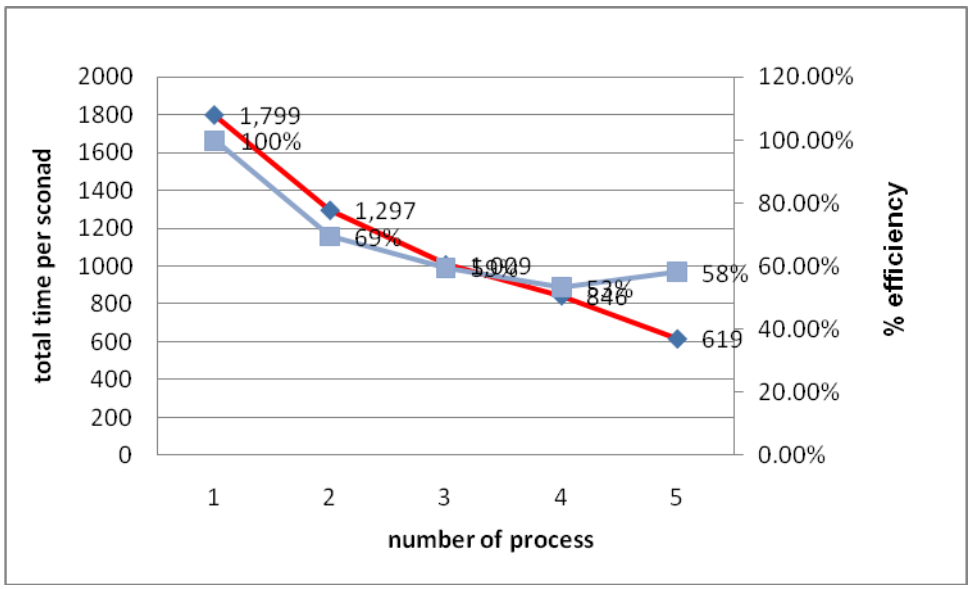
الگوریتم	Median	Smooth	Midpoint
زمان	1799.4	67.013	1807.8

جدول ۳-۲ زمان های اجرای سه الگوریتم با پنجره ۷*۷

این زمانها مبنای سنجش کارایی و افزایش سرعت در حالت موازی است.
هنگامی که پردازش بین دو پردازندها تقسیم می شود زمان اجرا به صورت جدول های زیر کاهش پیدا می کند .
با توجه به جدول و نمودار کارایی فیلتر Median کارایی این الگوریتم هنگامی که دو پردازنده در حال پردازش می باشند ۶۹٪ درصد می باشد و سرعت پردازش تا ۱,۴ افزایش میابد و حکایت از این موضوع دارد که این الگوریتم توانایی موازی سازی خوبی دارد .

median					
# process	computation time	communication time	total time	speed up	%efficiency
1	1798.190107	1.162985	1799.4	1.00	100.00%
2	1138.402126	158.519364	1296.9	1.39	69.37%
3	761.721208	247.486852	1009.2	1.78	59.43%
4	538.648932	307.531538	846.18	2.13	53.16%
5	442.601258	176.32817	618.93	2.91	58.14%

جدول ۳-۳ کارایی الگوریتم Median 7*7

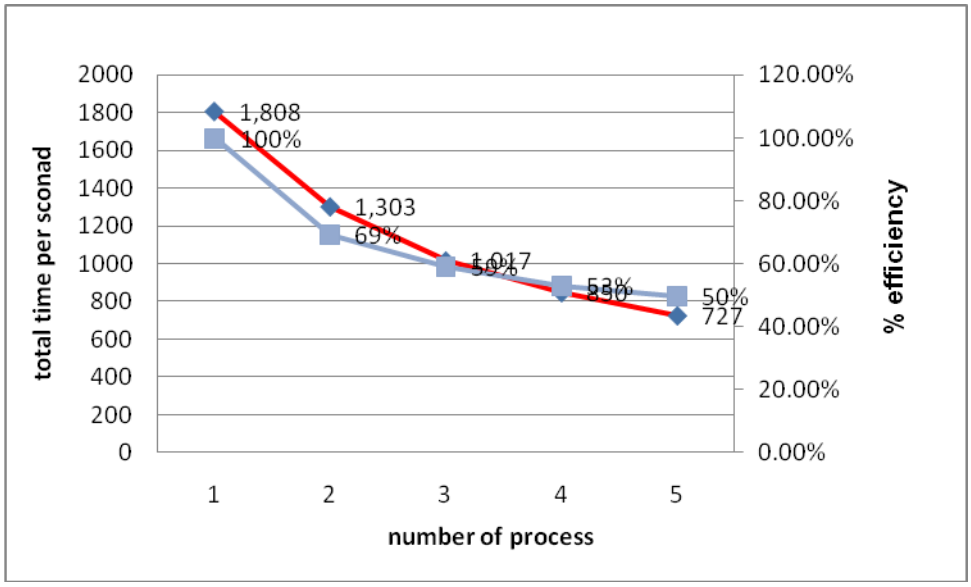


نمودار ۳-۱ کارایی الگوریتم Median 7*7

با توجه به جدول و نمودار کارایی فیلتر Midpoint کارایی این الگوریتم هنگامی که دو پردازنده در حال پردازش می باشند ۶۹٪ درصد می باشد و سرعت پردازش تا ۱,۳۹ افزایش میابد و حکایت از این موضوع دارد که این الگوریتم نیز توانایی موازی سازی خوبی دارد .

midpoint					
# process	computation time	communication time	total time	speed up	%efficiency
1	1806.624753	1.149436	1807.8	1.00	100.00%
2	1142.168774	160.992559	1303.2	1.39	69.36%
3	768.018604	248.959171	1017	1.78	59.25%
4	543.043199	306.507285	849.55	2.13	53.20%
5	432.152042	294.367671	726.52	2.49	49.77%

جدول۳-۴ کارایی الگوریتم Midpoint 7*7



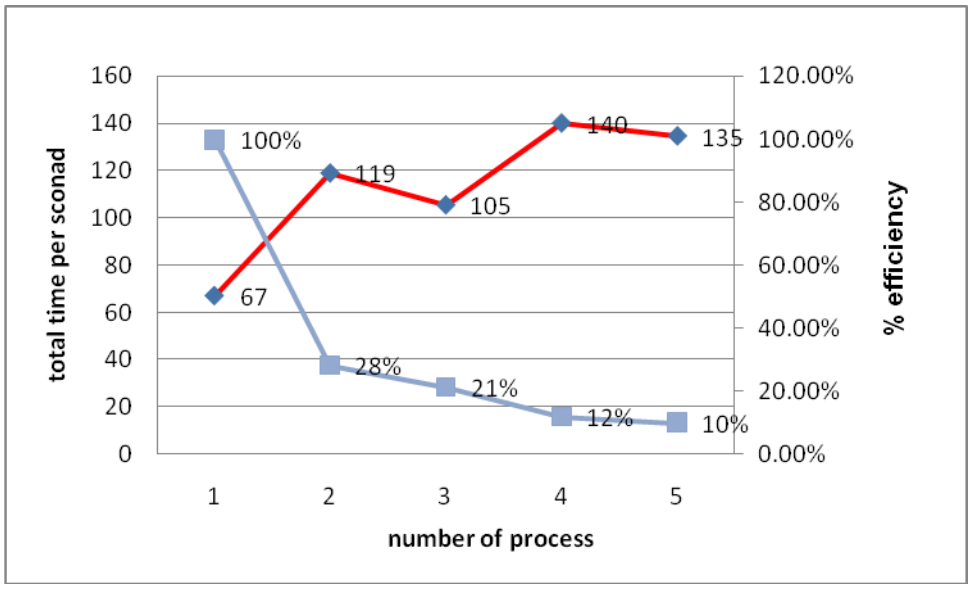
نمودار ۳-۲ کارایی الگوریتم Midpoint 7*7

با توجه به جدول و نمودار کارایی فیلتر Smooth کارایی این الگوریتم در هنگامی که دو پردازنده در حال پردازش می باشند کاهش پیدا می کند و این مشکل به دلیل حجم پردازش کم و همچنین زمان انتقال بالای اطلاعات می باشد که سربار ارسال و جمع آوری اطلاعات بیش از حد زیاد بوده و نشان دهنده این موضوع می باشد که این الگوریتم قابلیت موازی سازی خوبی ندارد.

smooth					
# process	computation time	communication time	total time	speed up	%efficiency
1	65.651854	1.361282	67.013	1	100.00%
2	33.699098	85.11592	118.82	0.5640123	28.20%
3	22.97776	82.389631	105.37	0.635995	21.20%
4	17.244943	122.702807	139.95	0.478844	11.97%
5	13.53493	121.0006841	134.54	0.498107	9.96%

جدول ۳-۵ کارایی الگوریتم Smooth 7*7

هنگامی که تعداد پردازشگر ها افزایش می یابد زمان بیشتری صرف ارسال جمع آوری و پردازش دوباره لبه ها می شود.



نمودار۳-۳ کارایی الگوریتم Smooth 7*7

۸-۳-2 الگوریتم های ۵*۵

زمان اجرای الگوریتم سریال حذف نویز از تصویر برای سه الگوریتم زیر برابر است با

الگوریتم	Median	Smooth	Midpoint
زمان	1214	53.502	1239.7

جدول ۳-۵ زمان های اجرای سه الگوریتم با پنجره ۵*۵

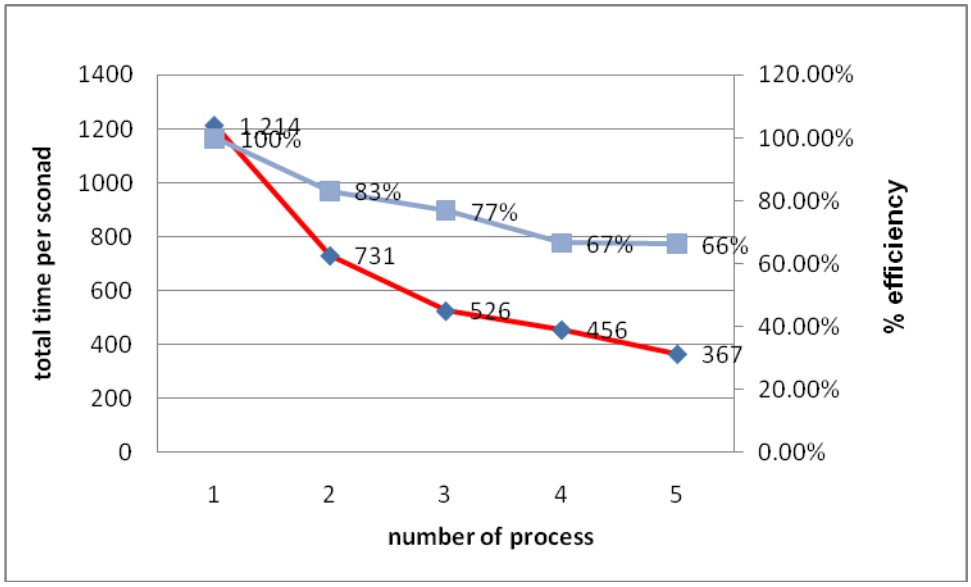
این زمانها مبنای سنجش کارایی و افزایش سرعت در حالت موازی است. همان گونه که از زمان پردازش پیداست زمان اجرا از الگوریتم های ۷*۷ کمتر می باشد

هنگامی که پردازش بین دو پردازندها تقسیم می شود زمان اجرا به صورت جدول های زیر کاهش پیدا می کند .

با توجه به جدول و نمودار کارایی فیلتر Median کارایی این الگوریتم هنگامی که دو پردازنده در حال پردازش می باشند ۸۳٪ درصد می باشد و سرعت پردازش تا ۱,۶۶ افزایش میابد و حکایت از این موضوع دارد که این الگوریتم توانایی موازی سازی خوبی دارد.

median					
# process	computation time	communication time	total time	speed up	% efficiency
1	1212.446777	1.598795	1214	1	100.00%
2	637.556164	93.6655434	731.22	1.66	83.01%
3	408.935809	117.447633	526.38	2.31	76.88%
4	299.732633	156.340009	456.07	2.66	66.55%
5	241.314201	125.303521	366.62	3.31	66.23%

جدول۳-۶ کارایی الگوریتم Median 5*5

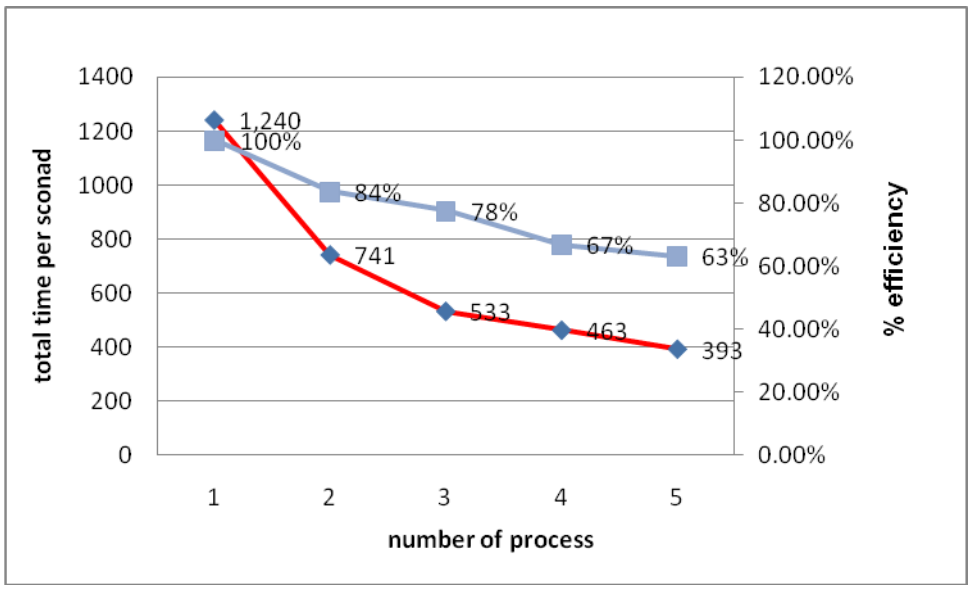


نمودار۳-۴ کارایی الگوریتم Median 5*5

با توجه به جدول و نمودار کارایی فیلتر Midpoint کارایی این الگوریتم هنگامی که دو پردازنده در حال پردازش می باشند ۶۹٪ درصد می باشد و سرعت پردازش تا ۱,۳۹ افزایش میابد و حکایت از این موضوع دارد که این الگوریتم نیز توانایی موازی سازی خوبی دارد .

midpoint					
# process	computation time	communication time	total time	speed up	% efficiency
1	1238.268682	1.41194	1239.7	1.00	100.00%
2	647.339385	94.127772	741.47	1.67	83.60%
3	414.233629	118.810327	533.04	2.33	77.52%
4	304.692238	158.328293	463.02	2.68	66.93%
5	242.87363	150.15547	393.03	3.15	63.08%

جدول۳-۷ کارایی الگوریتم Midpoint 5*5

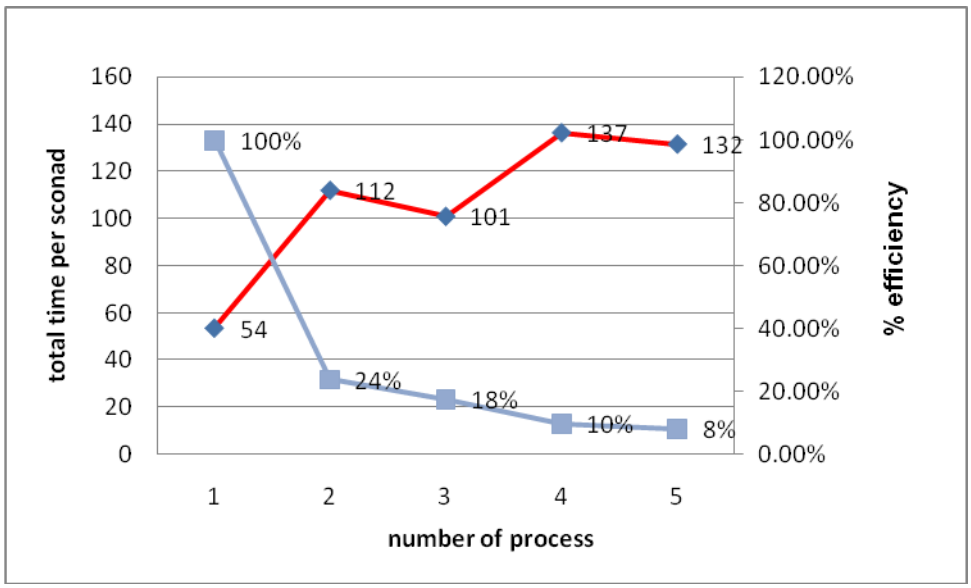


نمودار۳-۵ کارایی الگوریتم Midpoint 5*5

با توجه به جدول و نمودار کارایی فیلتر Smooth کارایی این الگوریتم در هنگامی که دو پردازنده در حال پردازش می باشند کاهش پیدا می کند و این مشکل به دلیل حجم پردازش کم و همچنین زمان انتقال بالای اطلاعات می باشد که سربار ارسال و جمع آوری اطلاعات بیش از حد زیاد بوده و نشان دهنده این موضوع می باشد که این الگوریتم قابلیت موازی سازی خوبی ندارد.

smooth					
# process	computation time	communication time	total time	speed up	% efficiency
1	52.147717	1.354624	53.502	1.00	100.00%
2	26.853739	85.191265	112.05	0.48	23.88%
3	18.226565	82.764304	100.99	0.53	17.66%
4	13.802712	122.711973	136.51	0.39	9.80%
5	10.813541	120.88083	131.69	0.41	8.13%

جدول۳-۸ کارایی الگوریتم Smooth 5*5



نمودار۳-۶ کارایی الگوریتم Smooth 5*5

۸-۳-3 الگوریتم های 3*3

زمان اجرای الگوریتم سریال حذف نویز از تصویر برای سه الگوریتم زیر برابر است با

الگوریتم	Median	Smooth	Midpoint
زمان	211.998	29.5482	227.473

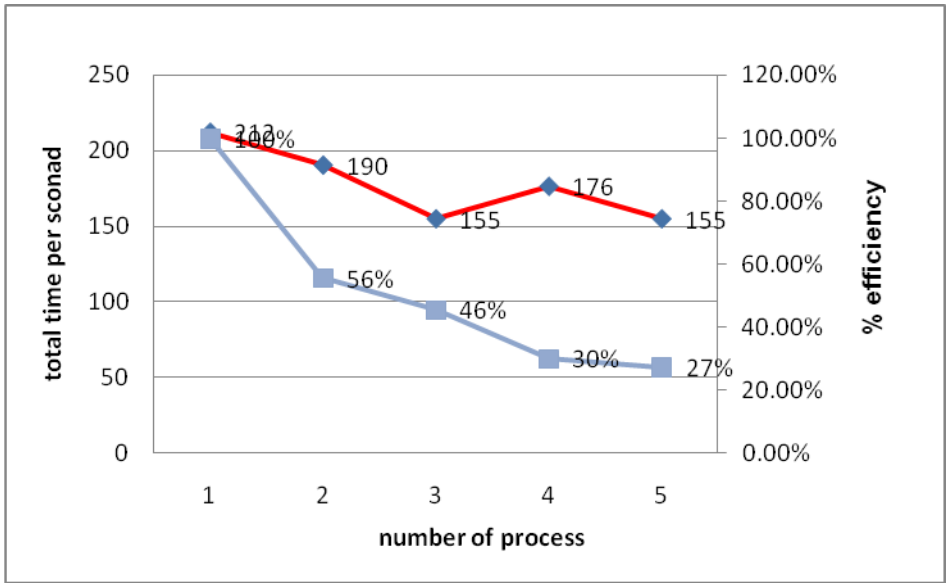
جدول ۴-۹ زمان های اجرای سه الگوریتم با پنجره ۳*۳

این زمانها مبنای سنجش کارایی و افزایش سرعت در حالت موازی است. همان گونه که از زمان پردازش پیداست زمان اجرا از الگوریتم های 5*5 و ۷*۷ کمتر می باشد. هنگامی که پردازش بین دو پردازندها تقسیم می شود زمان اجرا به صورت جدول های زیر کاهش پیدا می کند .

با توجه به جدول و نمودار کارایی فیلتر Median کارایی این الگوریتم هنگامی که دو پردازنده در حال پردازش می باشند 55.70٪ درصد می باشد و سرعت پردازش تا 1.11 افزایش میابد و با توجه به زمان محاسبات می توان مشاهده کرد که این الگوریتم توانایی موازی سازی خوبی دارد ولی نیاز به یک بستر ارتباطی کارامدتری دارد.

median					
# process	computation time	communication time	total time	speed up	%efficiency
1	210.621593	1.375986	211.998	1.00	100.00%
2	106.915683	83.380482	190.296	1.11	55.70%
3	69.145355	85.708262	154.854	1.37	45.63%
4	50.582119	125.722191	176.304	1.20	30.06%
5	40.995735	113.839511	154.835	1.37	27.38%

جدول۳-۱۰ کارایی الگوریتم Median 3*3

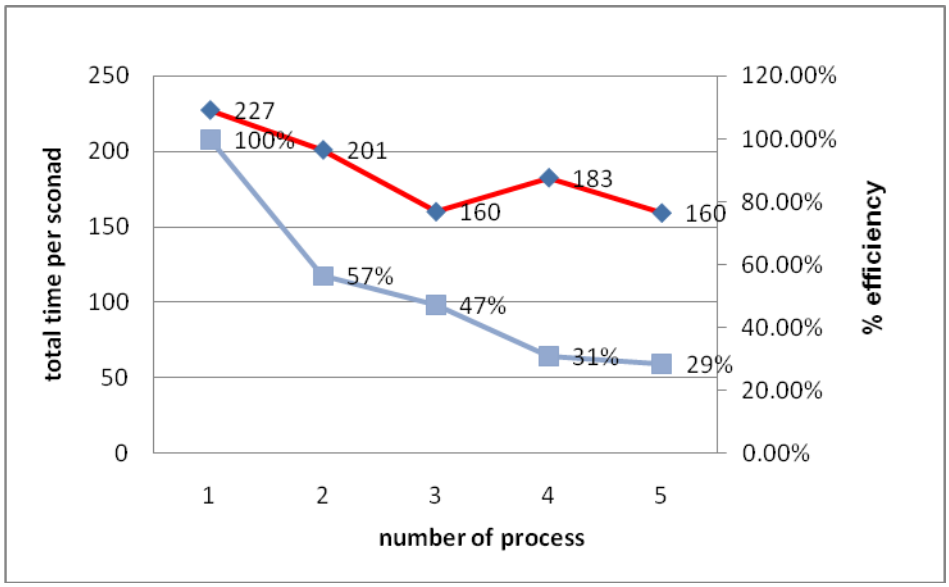


نمودار۳-۷ کارایی الگوریتم Median 3*3

با توجه به جدول و نمودار کارایی فیلتر Midpoint کارایی این الگوریتم هنگامی که دو پردازنده در حال پردازش می باشند 56.24٪ درصد می باشد و سرعت پردازش تا 1.12 افزایش میابد و با توجه به زمان محاسبات می توان مشاهده کرد که این الگوریتم توانایی موازی سازی خوبی دارد ولی نیاز به یک بستر ارتباطی کارامدتری دارد.

midpoint					
# process	computation time	communication time	total time	speed up	%efficiency
1	226.099754	1.372886	227.473	1.00	100.00%
2	118.026991	83.206586	201.234	1.13	56.52%
3	75.116523	85.312837	160.429	1.42	47.26%
4	55.729468	127.094998	182.824	1.24	31.11%
5	44.290373	115.286844	159.577	1.43	28.51%

جدول۳-۱۱ کارایی الگوریتم Midpoint 3*3

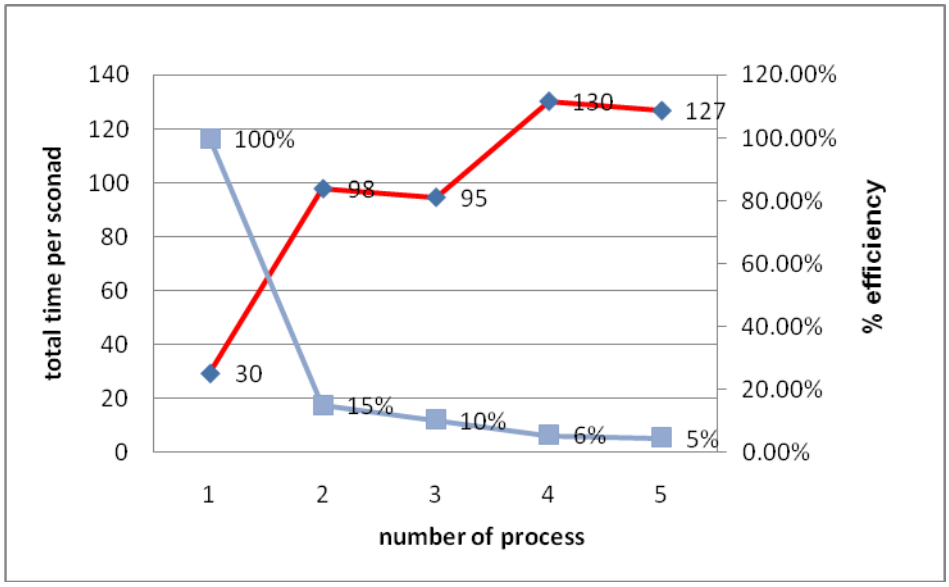


نمودار۳-۸ کارایی الگوریتم Midpoint 3*3

با توجه به جدول و نمودار کارایی فیلتر Smooth کارایی این الگوریتم در هنگامی که دو پردازنده در حال پردازش می باشند کاهش پیدا می کند و این مشکل به دلیل حجم پردازش کم و همچنین زمان انتقال بالای اطلاعات می باشد که سربار ارسال و جمع آوری اطلاعات بیش از حد زیاد بوده و نشان دهنده این موضوع می باشد که این الگوریتم قابلیت موازی سازی خوبی ندارد. حتی با یک بستر با کارامدی و قابلیت نقل و انتقال بالا نیز کارای نخواهد داشت.

smooth					
# process	computation time	communication time	total time	speed up	%efficiency
1	28.18355	1.364685	29.5482	1.00	100.00%
2	14.532747	83.513865	98.0466	0.30	15.07%
3	9.909227	84.92261	94.8318	0.31	10.39%
4	7.486619	122.911005	130.398	0.23	5.67%
5	5.88104	121.085023	126.966	0.23	4.65%

جدول۳-۱۲ کارایی الگوریتم Smooth 3*3



نمودار۳-۹ کارایی الگوریتم Smooth 3*3

ضمیمه ۱

مراحل نصب MPICH در محیط Linux

- کپی کردن فایل mpich-1.2.7 در دایرکتوری مورد نظر (مثلا /home).

```
$ gunzip –c mpich.tar.gz
$ tar –xvf mpich.tar
```

- در دایرکتوری mpich:

```
% ./configure –prefix=/usr/local/mpich-1.2.7 | tee c.log
```

- اضافه کردن دستور زیر در فایل .bashrc

```
export  PATH=/home/mpich-1.2.7/bin:$PATH
```

- نصب برنامه

```
$ make
# make install
```

- بررسی وضعیت نصب rsh

```
# rpm –q rsh rsh-server
        Rsh-0.17-25.3
        Rsh-server-0.17.25.3
```

- فعال کردن سرویس rsh

```
# su –
# chkconfig rsh on
# chkconfig rlogin on
# xinetd –restart
```

- اصلاح فایل /etc/hosts به صورت زیر

```
192.168.60.1  station0
192.168.60.2  station1
192.168.60.3  station2
…..
```

- اصلاح فایل /etc/hosts.equiv به صورت زیر

```
Station0
Station1
Station2
….
```

- ایجاد فایل /user/.rhosts.txt و تنظیمات آن

```
# touch /user/.rhosts.txt
# chmod 600 /user/.rhosts.txt
```

- اصلاح فایل /user/.rhosts.txt به صورت زیر

```
Station0
Station1
Station2
….
```

- بررسی عملکرد rsh: با rsh از هر نود به همه نودها متصل می شویم تا از عملکرد آن اطمینان حاصل شود.(البته باید سیستم ها دارای user name و passwordهای یکسان برای اجرای برنامه موازی باشند.)

```
$ rsh station0
```

برای اجرای یک برنامه ابتدا باید آنرا به کمک کمپایلر mpicc کامپایل کرده و فایل a.out را ایجاد می کنیم. این فایل باید در تمامی کامپیوترها در یک مسیر مشخص و ثابت کپی شود سپس با استفاده از دستور mpirun که در کامپیوتر اصلی اجرا می گردد برنامه به صورت همزمان بر روی کلیه کامپیوترها اجرا می گردد.

```
Mpicc pmedian.c
Mpirun –np 4 pmedian.out
```

Pmedian.c نام برنامه موازی شده و عدد ۴ تعدد پردازنده های مورد استفاده است.

منابع

- Advanced Linux Programming
- Beginning Linux Programming, 2nd Edition
- http://www.mhpcc.edu/training/workshop/mpi/MAIN.html
- Digital Signal and Image Processing 2nd Edition
- Digital Signal and Image Processing Using MATLAB
- http://www.imageprocessingplace.com
- http://www-unix.mcs.anl.gov
- http://runtime.futurs.inria.fr/mpi/
- http://www.bgl.mcs.anl.gov
- http://www.pgroup.com/resources/mpitools.htm