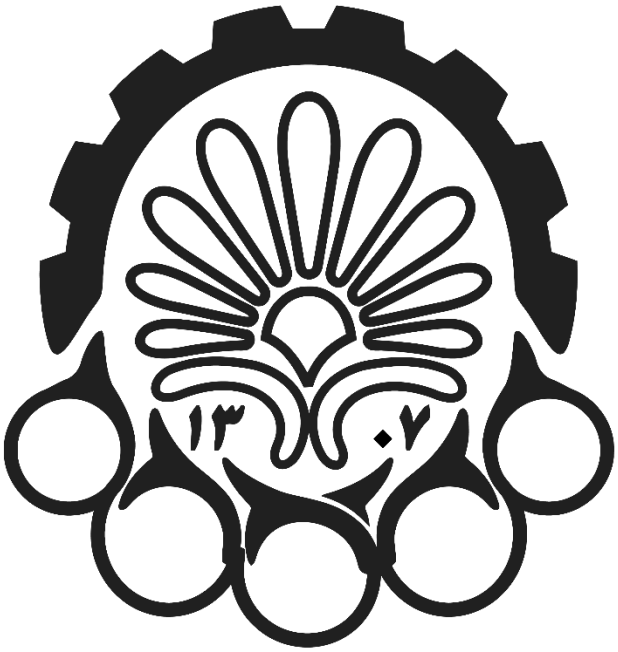
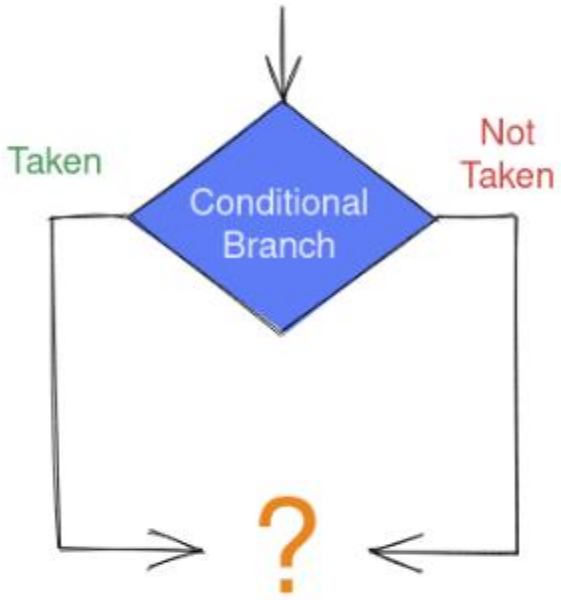
به نام خدا



دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

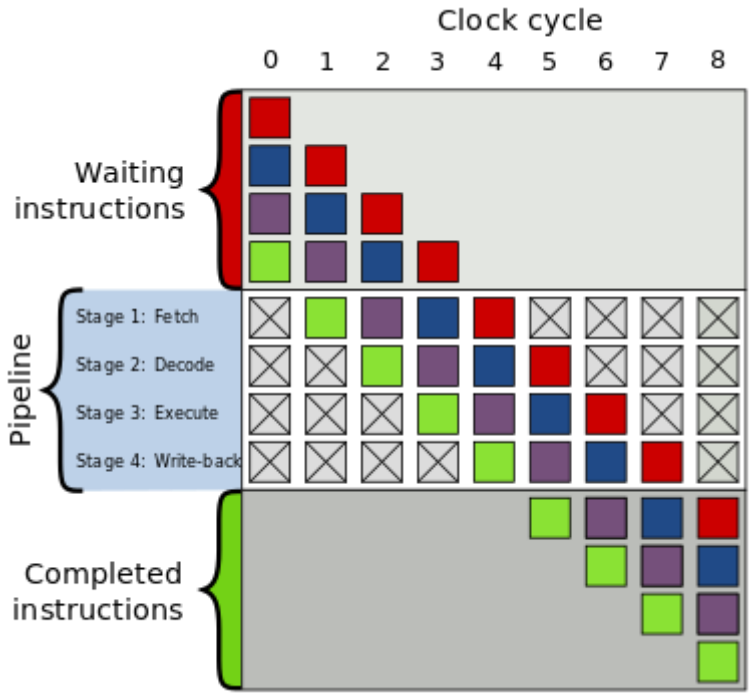


در معماری کامپیوتر، پیش بینی پرش یک مدار دیجیتال است که تلاش می‌کند حدس بزند که یک پرش چه راهی برای مثال ساختار-if then-elseمی‌رود قبل از اینکه به‌طور قطعی شناخته شود. مقصود از پیش بینی پرش بهبود جریان در دستورات خط لوله است. پیش بینی‌های پرش نقش اساسی در دست یابی به کارایی مؤثر بالا در خیلی از معماری‌های ریزپردازنده‌های جدید مثل x86 ایفا می‌کند.

منظور از پرش معمولا دستورات پرش شرطی است شرط پرش می‌تواند هم "not-taken" باشد و اجرا را با اولین پرش کد بلافاصله بعد از پرش شرطی است ادامه دهد یا می‌تواند " taken" شود و به مکان متفاوتی در حافظه برنامه پرش کند، جایی که دومین پرش کد ذخیره شده‌است. به‌طور دقیق مشخص نیست که آیا پرش شرطی taken یا not-taken خواهد شد تا وقتی که محاسبه شود و پرش شرطی از قسمت اجرا در خط لوله عبور می‌کند. (شکل یک) بدون پیش بینی پرش، پردازنده باید تا زمانی که دستور پرش شرطی از حالت اجرا عبور کند، قبل از اینکه دستور بعدی بتواند وارد حالت fetch در خط لوله شود، صبر کند. پیش بینی پرش تلاش می‌کند تا از اتلاف زمان اجتناب کند با تلاش کردن اینکه حدس بزند که آیا پرش شرطی احتمال دارد taken یا not taken شود. پرش که حدس زده شود به احتمال زیاد fetch می‌شود و به صورت حدسی اجرا می‌شود. اگر بعداً مشخص شود که حدس غلط بوده سپس اجرای حدسی یا اجرای بخشی از دستورات اجرا شده نادیده گرفته می‌شوند و خط لوله دوباره شروع می‌کند با پرش صحیح، تحمیل تأخیر

زمانی که در رابطه با پیش بینی غلط به هدر می‌رود برابر است با تعدادی از مراحل در خط لوله. از مرحلهٔ fetch تا مرحلهٔ اجرا. ریزپردازنده‌های جدید تمایل دارند که خط لوله‌های نسبتاً طولانی داشته باشند به‌طوری‌که پیش بینی غلط تأخیر بین ۱۰ تا ۲۰ دوره ساعتی اتفاق می‌افتد. در نتیجه، درست کردن خط لوله طولانی‌تر، نیاز پیش بینی پرش پیشرفته تر را افزایش می‌دهد.

اولین زمان که به دستور پرش مواجه شد، اطلاعات زیادی بر پایهٔ پیش بینی نیست. اما پیش بینی پرش اطلاعات را نگهداری می‌کند چه پرش‌ها taken شوند یا نشوند. وقتی که با یک پرش شرطی که چندین بار با آن برخورد کرده مواجه می‌شود می‌تواند پیش بینی را بر پایهٔ تاریخ قرار دهد. پیش بینی پرش ممکنه برای مثال، تشخیص دهد که پرش شرطی اغلب انجام شده یا خیر، یا اینکه یک در میان انجام شده.



شکل ۱: نمونه‌ای از ۴-مرحله خط لوله است. جعبه‌های رنگی نشان دهنده دستورالعمل‌ها مستقل از یکدیگر

Static prediction

- Static branch predictions: they don't change

- Predict **always not taken**
- Predict **always taken**
  - Depends on the code
- Think about loops:
  - The branch at the end almost always jumps back (the loop repeats many times)
  - How about: **Backwards-Taken, Forward-Not-Taken**? (BTFNT)
- Can the **compiler** help?
  - Sure, if it can identify the likely branches. (E.g., loops or error checks)

پیش بینی پرش ایستا در معماری رایانه

پیش بینی پرش ایستا تکنیک ساده‌ای است که در معماری رایانه برای پیش بینی نتیجه‌ی پرش‌ها مانند ساختارهای "if-then-else" استفاده می‌شود بدون آنکه نیاز به بررسی پویای تاریخچه‌ی اجرای کد باشد. در عوض، نتیجه‌ی پرش را تنها بر اساس خود دستورالعمل پرش پیش بینی می‌کند.

در معماری رایانه، پیش بینی پرش مدار دیجیتالی است که سعی می‌کند قبل از مشخص شدن مسیر یک پرش مثلاً ساختار-if-then" else آن را حدس بزند. هدف پیش بینی پرش، بهبود جریان در پایپ لاین دستورالعمل است.

در مورد پیش بینی پرش ایستا، سخت‌افزار زیربنایی فرض می‌کند که یا پرش همیشه گرفته نمی‌شود (not taken)یا همیشه گرفته می‌شود(taken). برای مثال، کد زیر را در نظر بگیرید:

```
int a=0;

while (a<5) {

دستورالعمل پرش، شرط می‌تواند درست یا نادرست باشد

  if (a%2==0) {.....}

  a++;

}
```

در این حالت، تکنیک پیش بینی پرش ایستا براساس اینکه شرط "if" درست یا نادرست است، یک پیش بینی انجام می‌دهد.

هنگامی که برای اولین بار با دستورالعمل پرش شرطی مواجه می‌شویم، اطلاعات زیادی برای پایه‌گذاری یک پیش بینی وجود ندارد. اما پیش بینی پرش سابقه‌ی taken یا not taken پرش‌ها را ثبت می‌کند. هنگامی که با پرش شرطی‌ای مواجه می‌شود که قبلاً چندین بار دیده شده، می‌تواند پیش بینی خود را بر اساس این سابقه انجام دهد.

با این حال، مهم است که توجه داشته باشیم که پیش بینی ایستا ساده‌تر از پیش بینی پویا است. همچنین، کامپایلر می‌تواند بر اساس تحلیل یا اطلاعات پروفایل تعیین کند که آیا احتمالاً یک پرش گرفته شود یا نه.

برخی پردازنده‌ها اجازه می‌دهند که پیش بینی پرش به قرار دادن در کد به منظور ایمکه مشخص شود که آیا پیش بینی ایستا باید taken یا not taken شود اشاره بکند Pentium 4 .تلاش می‌کند پیش بینی پرش اشاره بکند در حالی که این مشخصه در پردازنده‌های اخیر کمیاب است .

ویدیو آموزشی پیش بینی کننده ایستا

Next line precdictor

NLPیک تکنیک پیش بینی پرش است که کارایی دریافت دستورالعمل ها را بهبود می بخشد.

برخلاف پیش بینی کننده های پیچیده تر، NLPروی پیش بینی اجرای یا عدم اجرای دستورالعمل بعدی (یعنی دستورالعمل بلافاصله پس از دستورالعمل فعلی) تمرکز می کند.

این پیش بینی در سطح دستورالعمل عمل می کند تا توقف های خط لوله ناشی از پیش بینی های نادرست را به حداقل برساند.

نحوه عملکرد:NLP

NLPبه داده های تاریخی گسترده یا ماشین های حالت پیچیده متکی نیست.

در عوض، از قواعد ساده ای بر اساس الگوهای مشاهده شده در طول اجرا استفاده می کند.

نحوه کار معمول آن به این صورت است:

هنگام دریافت دستورالعمل، NLPپیش بینی می کند که آیا دستورالعمل بعدی اجرا خواهد شد یا خیر.

اگر پیش بینی درست باشد، خط لوله به طور روان ادامه می یابد.

اگر پیش بینی نادرست باشد (به عنوان مثال به دلیل یک انشعاب)، خط لوله به طور مختصری متوقف می شود تا مسیر صحیح تعیین شود.

محدودیت ها:

NLPبرای الگوهای کنترل جریان ساده موثر است، اما ممکن است با پرش های پیچیده تر مشکل داشته باشد.

اهداف پرش یا رفتار بلندمدت را در نظر نمی گیرد.

اگر برنامه رفتار پرش ای پیچیده ای داشته باشد، ممکن است سایر پیش بینی کننده ها) مانند BTB یا پیش بینی کننده های تورنمنت (دقیق تر باشند.

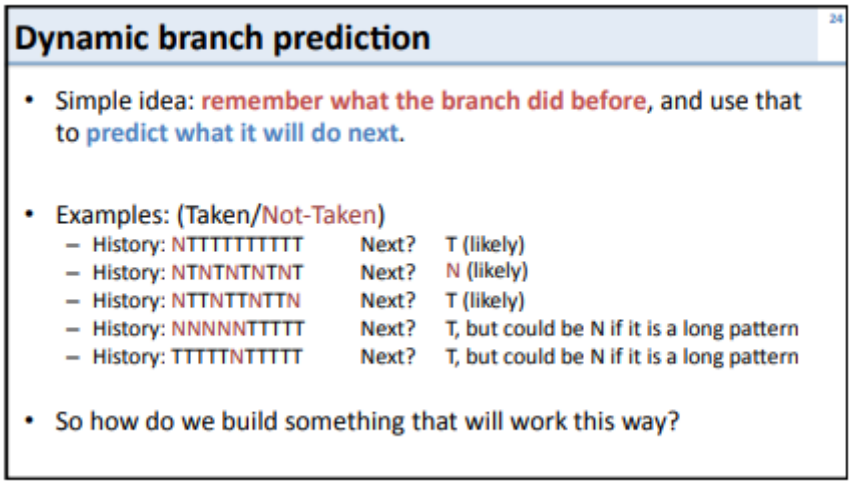
ملاحظات:

NLPبا قربانی کردن دقت، سرعت را به دست می آورد.

در حالی که همیشه عالی نیست، تاخیر کم آن را برای حفظ یک خط لوله دستورالعمل به خوبی تغذیه شده ارزشمند می کند.

به طور خلاصه، NLPیک پیش بینی کننده پرش سبک و سریع تصمیم گیری است که به پردازنده کمک می کند به طور کارآمد کار کند.

Dynamic branch predictor



پیش بینی‌کننده‌های پرش پویا با تحلیل پویای رفتار پرش‌ها در حین اجرای برنامه، برای پیش بینی نتایج آینده‌ی آن‌ها کار می‌کنند .در اینجا به طور معمول نحوه‌ی عملکرد آن‌ها را توضیح می‌دهیم:

ثبت‌کننده‌ی تاریخچه :پیش بینی‌کننده یک ثبت‌کننده‌ی تاریخچه یا بافر حفظ می‌کند که نتایج دستورات پرش اخیر را ثبت می‌کند . این ثبت‌کننده دنباله‌ای از بیت‌ها را ذخیره می‌کند که هر بیت نشان‌دهنده‌ی نتیجه‌ی (گرفته شده یا نگرفته شده) یک پرش گذشته است.

تشخیص الگو :پیش بینی‌کننده برای پیش بینی پرش‌های آینده، الگوهای موجود در تاریخچه‌ی پرش را تجزیه و تحلیل می‌کند .ممکن است از تکنیک‌های مختلفی برای شناسایی الگوهای تکرارشونده یا همبستگی بین نتایج پرش‌های گذشته و رفتار آینده استفاده کند.

مکانیزم پیش بینی :بر اساس الگوهای مشاهده شده در تاریخچه‌ی پرش، پیش بینی‌کننده از یک مکانیزم پیش بینی برای تخمین احتمال گرفته شدن یا نگرفته شدن یک پرش استفاده می‌کند .این مکانیزم می‌تواند ساده باشد، مانند استفاده از جستجوی جدولی بر اساس تاریخچه‌ی پرش، یا پیچیده‌تر باشد، مانند استفاده از الگوریتم‌های یادگیری ماشین مانند شبکه‌های عصبی یا پرسپترون‌ها که در ادامه درباره آن بحث می کنیم.

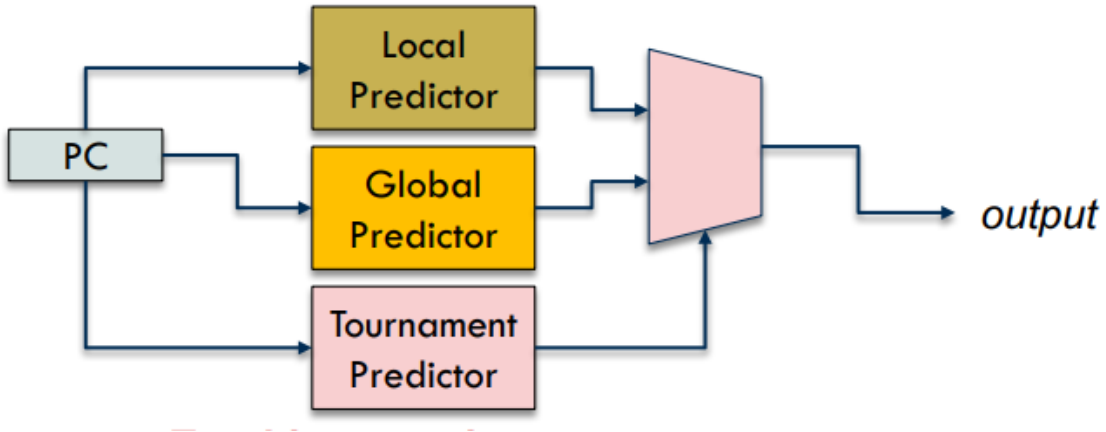
سازگاری :پیش بینی‌کننده‌های پویا به طور مداوم مکانیزم‌های پیش بینی خود را بر اساس بازخورد از نتایج واقعی پرش‌ها تطبیق و به روز رسانی می‌کنند .اگر یک پیش بینی نادرست باشد، پیش بینی‌کننده وضعیت داخلی خود را برای بهبود پیش بینی‌های آینده تنظیم می‌کند . این فرآیند سازگاری به پیش بینی‌کننده اجازه می‌دهد تا از رفتار گذشته بیاموزد و با تغییرات در الگوهای اجرای برنامه سازگار شود.

سطوح مختلف پیش بینی :بسیاری از پیش بینی‌کننده‌های پویا از چندین سطح پیش بینی برای درک انواع مختلف الگوهای رفتار پرش استفاده می‌کنند .به عنوان مثال، آن‌ها ممکن است شامل پیش بینی‌کننده‌های سراسری که رفتار کلی برنامه را تجزیه و تحلیل می‌کنند، و همچنین پیش بینی‌کننده‌های محلی که روی مناطق خاصی از کد تمرکز می‌کنند.

ترکیب :پیش بینی‌کننده‌های پویا ممکن است برای تکمیل تحلیل پویای خود، استراتژی‌های پیش بینی دیگری مانند پیش بینی‌کننده‌های ایستا یا پیش بینی‌کننده‌های مبتنی بر تاریخچه را نیز در خود ادغام کنند .این ترکیب به پیش بینی‌کننده اجازه می‌دهد تا از نقاط قوت رویکردهای مختلف پیش بینی استفاده کند و دقت کلی پیش بینی را بهبود بخشد.

به طور کلی، پیش بینی‌کننده‌های پرش پویا با تجزیه و تحلیل رفتار پرش‌ها در زمان واقعی، پیش بینی‌های دقیقی در مورد پرش‌های آینده انجام می‌دهند و استراتژی‌های خود را بر اساس الگوهای مشاهده شده و بازخورد از پیش بینی‌های گذشته تطبیق می‌دهند .این رویکرد پویا و تطبیقی با کاهش تعداد پرش‌های پیش بینی‌شده‌ی نادرست و بهینه‌سازی اجرای برنامه، به بهبود عملکرد کمک می‌کند.

Tournament Branch Predictor



اجزای تشکیل دهنده ی پیش بینی کننده این پیش بینی کننده از چندین پیش بینی کننده ی کوچکتر تشکیل شده است .به طور معمول، دو نوع پیش بینی کننده وجود دارد :یک پیش بینی کننده ی محلی و یک پیش بینی کننده ی global

پیش بینی کننده ی محلی :این جزء، نتیجه ی پرش را بر اساس تاریخچه ی آن پرش خاص پیش بینی می کند .این جزء یک بافر یا جدول کوچک را حفظ می کند که نتایج اجرای های قبلی همان پرش را ثبت می کند .این پیش بینی کننده ممکن است از تکنیک هایی مانند جدول تاریخچه ی الگو (PHT) یا ماشین حالت محدود (FSM) برای پیش بینی نتایج استفاده کند.

پیش بینی کننده ی global این جزء، نتیجه ی پرش را بر اساس رفتار کلی پرش ها در سراسر برنامه پیش بینی می کند .این جزء الگوهای مختلف پرش ها را مشاهده می کند و از آنها برای ایجاد پیش بینی استفاده می کند .این پیش بینی کننده ممکن است از تکنیک هایی مانند یک ثبتگر تاریخچه ی global (GHR) به همراه یک جدول الگو استفاده کند.

سازوکار انتخاب :خروجی های هر دو پیش بینی کننده ی محلی و global با استفاده از یک سازوکار انتخاب ترکیب می شوند .این سازوکار می تواند به سادگی انتخاب یک پیش بینی کننده بر اساس عملکرد گذشته باشد، یا می تواند پیچیده تر باشد، مانند استفاده از یک پیش بینی کننده ی فرا (meta-predictor) برای پیش بینی اینکه کدام پیش بینی کننده احتمالاً برای یک پرش خاص بهتر عمل می کند.

سازوکار تورنمنت :سازوکار انتخاب اغلب به عنوان یک "تورنمنت" پیاده سازی می شود که در آن دو پیش بینی کننده با هم رقابت می کنند .برنده ی این تورنمنت، یعنی پیش بینی کننده ای که برای یک پرش ی خاص دقیق تر در نظر گرفته می شود، برای انجام پیش بینی نهایی انتخاب می شود.

سازوکار به روز رسانی :پس از شناخته شدن نتیجه ی پرش، پیش بینی کننده ها به طور مناسب به روز می شوند .این سازوکار به روز رسانی تضمین می کند که پیش بینی کننده ها در طول زمان با تغییرات رفتاری برنامه سازگار شوند .به عنوان مثال، اگر پیش بینی یک پیش بینی کننده درست بود، وضعیت داخلی آن ممکن است به گونه ای تنظیم شود که آن پیش بینی را در آینده ترجیح دهد.

پیش بینی کننده ی پرش ای تورنمنت با ترکیب نقاط قوت پیش بینی کننده های محلی و global، به دقت بالاتری در پیش بینی نتایج پرش ها دست می یابد و در نتیجه با کاهش تأخیرهای خط لوله ناشی از پیش بینی های نادرست پرش، عملکرد پردازنده را بهبود می بخشد .این پیش بینی کننده ها اجزای حیاتی پردازنده های مدرن هستند و به طور قابل توجهی در کارایی و عملکرد کلی آنها نقش دارند.

هدف از آوردن نمونه های بالا آشنایی با چند نمونه پیش بینی کننده ساده بود بقیه پیشبنی کننده ها از جمله,local globale, که در طول درس آمده برسی نمی شود و به سراغ چند نمونه که در طول درس به آن ها اشاره نشده از جمله TAGE و perceptron می پردازیم .

ویدیو آموزشی tournument

فایل جهت مرور بر مباحث پیش بینی کنند های local و global

TAGE

پیش بینی‌کننده‌ی پرش TAGE توسط André Seznec و Pierre Michaud به عنوان بهترین پیش بینی‌کننده‌ی پرش تا به امروز شناخته می‌شود و در دو مسابقه‌ی اخیر پیش بینی‌کننده‌ی پرش CBP2وCBP3 برنده شده است .این پیش بینی‌کننده در مقاله‌ای در سال ۲۰۰۶ معرفی شد:

A case for (partially) tagged Geometric History Length Branch Prediction- André Seznec, Pierre Michaud- Journal of Instruction Level Parallelism (JILP)۲۰۰۶

TAGEیک تکنیک پیشرفته برای بهبود دقت پیش بینی پرش است که در پردازنده‌های مدرن استفاده می‌شود. این تکنیک با مدیریت مجموعه‌ای از جداول به نام "جداول برچسب‌زده" که هر کدام طول تاریخچه‌ی هندسی متفاوتی را نشان می‌دهند، کار می‌کند.

عملکرد TAGE به صورت دقیق‌تر:

ثبت‌کننده‌ی تاریخچه TAGE :یک ثبت‌کننده‌ی تاریخچه‌ی کلی را حفظ می‌کند که نتایج پرش‌های اخیر را ثبت می‌کند. این ثبت‌کننده به عنوان یک شاخص برای جستجو در جداول‌های برچسب‌زده استفاده می‌شود.

جداول برچسب‌زده TAGE :از چندین جدول تشکیل شده که به طور معمول با نام T0, T1, T2, ..., Tk شناخته می‌شوند. هر جدول با طول تاریخچه‌ی هندسی متفاوتی مطابقت دارد. به عنوان مثال، T0ممکن است طول تاریخچه‌ی کوتاهی داشته باشد (مثلاً 3 بیت)، در حالی که Tk ممکن است طول تاریخچه‌ی طولانی‌تری داشته باشد (مثلاً 12 بیت). این جداول پیش بینی‌هایی را برای پرش‌ها بر اساس تاریخچه‌ی آن‌ها ذخیره می‌کنند.

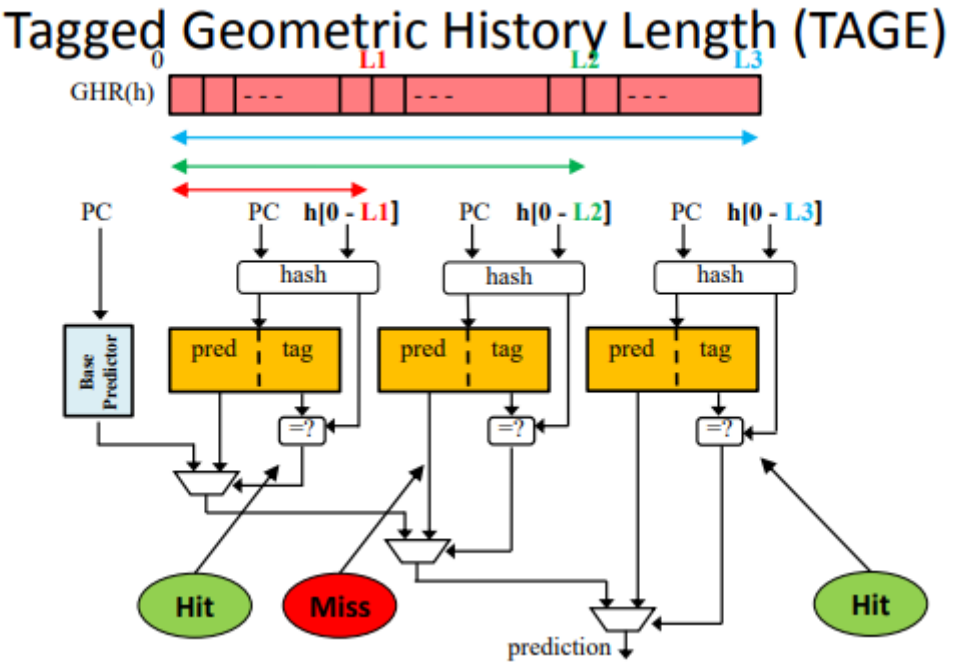
ایجاد برچسب :هنگامی که یک پرش رخ می‌دهد، از ثبت‌کننده‌ی تاریخچه برای ایجاد یک برچسب استفاده می‌شود. این برچسب اساساً یک نمایش متراکم از تاریخچه‌ی پرش است که برای جستجو در جداول‌های برچسب‌زده استفاده می‌شود.

پیش بینی TAGE :بر اساس برچسب ایجاد شده، پیش بینی‌هایی را از چندین جدول برچسب‌زده بازیابی می‌کند. پیش بینی‌های هر جدول با استفاده از یک طرح وزنی ترکیب می‌شوند، به طوری که وزن بیشتری به پیش بینی‌های جداول‌هایی با طول تاریخچه‌ی طولانی‌تر داده می‌شود. این به TAGE کمک می‌کند تا با الگوهای مختلف رفتار پرش سازگار شود.

پیش بینی نهایی :پیش بینی نهایی بر اساس پیش بینی‌های ترکیبی از تمام جداول‌های برچسب‌زده ایجاد می‌شود. به طور معمول، پیش بینی با بالاترین اعتماد (مثلاً پیش بینی از جدولی با طولانی‌ترین طول تاریخچه) به عنوان پیش بینی نهایی انتخاب می‌شود.

سازگاری TAGE :طول تاریخچه و سایر پارامترها را بر اساس دقت پیش بینی‌های گذشته به طور پویا تنظیم می‌کند. اگر یک پیش بینی نادرست باشد، TAGEوضعیت داخلی خود را به روز می‌کند تا پیش بینی‌های آینده را بهبود بخشد.

در کل، قدرت TAGE در توانایی آن برای ثبت و سازگاری با الگوهای مختلف رفتار پرش با حفظ چندین پیش بینی‌کننده با طول‌های تاریخچه‌ی متفاوت است. این سازگاری به TAGE کمک می‌کند تا در مقایسه با تکنیک‌های ساده‌تر پیش بینی پرش، به دقت بالاتری دست یابد.



فرض کنید حلقه‌ای در برنامه‌ای با رفتار پرش زیر داریم:

تکرار 1 :پرش می‌کند

تکرار 2 :پرش نمی‌کند

تکرار 3 :پرش می‌کند

تکرار 4 :پرش نمی‌کند

تکرار 5 :پرش می‌کند

حالا فرض کنید TAGE سه جدول را نگهداری می‌کندT0 :، T1وT2 ، با طول‌های تاریخچه به ترتیب 1، 2 و 3.

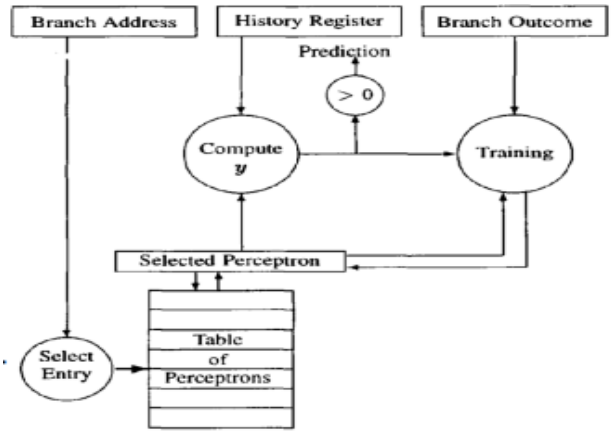
جدول) T0 تاریخچه 1 بیتی :(این جدول فقط خروجی پرش آخر را به خاطر می‌سپارد .با توجه به تاریخچه‌ی اخیر، ممکن است پرش بعدی را پرش می‌کند پیش بینی کند.

جدول) T1 تاریخچه 2 بیتی :(این جدول خروجی‌های دو پرش اخیر را به خاطر می‌سپارد .ممکن است الگوی متناوب را تشخیص دهد و پرش بعدی را پرش نمی‌کند پیش بینی کند.

جدول) T2 تاریخچه 3 بیتی :(این جدول خروجی‌های سه پرش اخیر را به خاطر می‌سپارد .ممکن است الگوی" پرش می‌کند، پرش نمی‌کند، پرش می‌کند "را تشخیص دهد و پرش بعدی را پرش می‌کند پیش بینی کند.

اکنون، TAGEپیش بینی‌ها را از هر سه جدول ترکیب می‌کند .از آنجایی که T2 طولانی‌ترین تاریخچه را دارد و یک الگو را تشخیص داده است، پیش بینی آن ممکن است وزن بیشتری داشته باشد .بنابراین، TAGEممکن است پیش بینی کند که پرش بعدی پرش می‌کند.

Perceptron branch predictor



پیش بینی کننده‌ی پرش بر پایه ی پرسپترون نوعی پیش بینی کننده است که از یک الگوریتم یادگیری ماشینی به نام پرسپترون برای پیش بینی کردن نتیجه‌ی پرش‌های شرطی در یک برنامه استفاده می‌کند. نحوه‌ی کارکرد آن به شرح زیر است:

آغازین‌سازی :پیش بینی کننده‌ی پرش بر پایه ی پرسپترون مجموعه‌ای از پرسپترون‌ها را آغازین‌سازی می‌کند. هر پرسپترون مربوط به یک ترکیب منحصربه‌فرد از بیت‌های تاریخچه‌ی پرش است.

تاریخچه‌ی پرش :هنگامی که در طول اجرای برنامه با دستور پرش مواجه می‌شویم، پیش بینی کننده‌ی پرش از یک ثبت‌کننده‌ی تاریخچه‌ی با اندازه ثابت برای ثبت کردن تاریخچه‌ی اخیر نتایج پرش استفاده می‌کند. هر بیت در ثبت‌کننده‌ی تاریخچه، نتیجه‌ی (گرفته شده یا نگرفته شده) یک پرش اخیر را نشان می‌دهد.

بردار ویژگی :ثبت‌کننده‌ی تاریخچه به عنوان یک بردار ویژگی عمل می‌کند که به عنوان ورودی به پرسپترون‌ها داده می‌شود. هر پرسپترون دارای وزن‌هایی مرتبط با ورودی‌های خود (بیت‌های تاریخچه‌ی پرش) است.

پیش بینی :پرسپترون مجموع وزنی ورودی‌های خود (بیت‌های تاریخچه) را محاسبه می‌کند و با اعمال یک تابع آستانه، پیش بینی‌ای را تولید می‌کند. اگر مجموع وزنی از آستانه‌ی مشخصی بیشتر شود، پرسپترون پیش بینی می‌کند که پرش گرفته می‌شود؛ در غیر این صورت، پیش بینی می‌کند که پرش گرفته نمی‌شود.

به‌روزرسانی وزن‌ها :پس از اینکه نتیجه‌ی پرش مشخص شد، پیش بینی کننده‌ی پرسپترون وزن‌های پرسپترون‌ها را بر اساس خطای پیش بینی به‌روزرسانی می‌کند. این فرآیند به‌روزرسانی به پیش بینی کننده‌ی پرسپترون کمک می‌کند تا با گذشت زمان با تغییرات رفتار پرش سازگار شود.

چندین پرسپترون :پیش بینی کننده‌ی پرش بر پایه ی پرسپترون به طور معمول چندین پرسپترون را حفظ می‌کند که هر کدام برای تشخیص الگوهای مختلف در تاریخچه‌ی پرش آموزش‌دیده‌اند. با ترکیب پیش بینی‌ها از چندین پرسپترون، پیش بینی کننده می‌تواند در پیش بینی کردن نتایج پرش به دقت بالاتری دست یابد.

به طور خلاصه، پیش بینی کننده‌ی پرش بر پایه ی پرسپترون از مجموعه‌ای از پرسپترون‌های آموزش‌دیده روی تاریخچه‌ی پرش برای پیش بینی کردن نتیجه‌ی پرش‌های شرطی استفاده می‌کند. این پیش بینی کننده به طور مداوم وزن‌های پرسپترون‌ها را بر اساس خطاهای پیش بینی به‌روزرسانی می‌کند و به آن امکان می‌دهد تا با تغییرات رفتار برنامه سازگار شود و به مرور زمان دقت پیش بینی را بهبود بخشد.

ویدیو آموزشی پیش بینی کننده با شبکه پرسپترون

مقاله مرتبط

Agree predictor

E. Sprangle, R. S. Chappell, M. Alsup, and Y. N. Patt. The Agree Predictor: A Mechanism for Reducing Negative Branch History Interference. ISCA-24, June 1997.

پیش بینی‌کننده‌ی Agree نوعی پیش بینی‌کننده‌ی پرش‌ی پویا است که با استفاده از توافق بین مجموعه‌ای از پیش بینی‌کننده‌های مجزا ، به دنبال بهبود دقت پیش بینی است .در اینجا نحوه‌ی کارکرد آن توضیح داده شده است:

مجموعه‌ای از پیش بینی‌کننده‌های مجزا :پیش بینی‌کننده‌ی Agree از چندین جزء پیش بینی تشکیل شده است که هر کدام از یک استراتژی پیش بینی یا طول تاریخچه‌ی متفاوتی استفاده می‌کنند .این اجزا می‌توانند شامل پیش بینی‌کننده‌های محلی، پیش بینی‌کننده‌های سراسری، پیش بینی‌کننده‌های شبکه‌ی عصبی یا هر مکانیزم پیش بینی دیگری باشند.

توافق پیش بینی :هنگامی که در حین اجرای برنامه با یک پرش مواجه می‌شویم، هر جزء پیش بینی مستقل از دیگری پیش بینی خود را برای نتیجه‌ی پرش (گرفته شده یا نگرفته شده) ایجاد می‌کند .سپس پیش بینی‌کننده این پیش بینی‌ها را با هم مقایسه می‌کند تا ببیند آیا بین اجزا توافق وجود دارد یا خیر.

تصمیم مبتنی بر اجماع :اگر سطح بالایی از توافق بین اجزای پیش بینی وجود داشته باشد (مثلاً اگر اکثر اجزا نتیجه‌ی یکسانی را پیش بینی کنند)، پیش بینی‌کننده‌ی Agree این پیش بینی اجماع را به عنوان پیش بینی نهایی برای پرش انتخاب می‌کند .این تصمیم مبتنی بر اجماع به بهبود دقت پیش بینی کمک می‌کند، زیرا این تصمیم منعکس‌کننده‌ی بینش‌های ترکیبی چندین استراتژی پیش بینی است.

مدیریت عدم توافق :در مواردی که اجزای پیش بینی با هم موافق نیستند (مثلاً اگر پیش بینی‌ها به طور مساوی تقسیم شده باشند یا اجماع مشخصی وجود نداشته باشد)، پیش بینی‌کننده‌ی Agree ممکن است از مکانیزم‌های دیگری برای حل عدم توافق استفاده کند .این می تواند شامل استفاده از یک استراتژی پیش بینی پشتیبان، تنظیم پویای وزن‌های پیش بینی یا استفاده از یک مکانیزم تصمیم‌گیری برای شکستن تساوی باشد.

سازگاری و یادگیری :مانند سایر پیش بینی‌کننده‌های پویا، پیش بینی‌کننده‌ی Agree به طور مداوم با نتایج پیش بینی‌های گذشته سازگار می‌شود و از آن‌ها یاد می‌گیرد .اگر یک پیش بینی نادرست باشد، پیش بینی‌کننده وضعیت داخلی خود را به‌روزرسانی می‌کند یا وزن‌های اجزای پیش بینی خود را برای بهبود پیش بینی‌های آینده تنظیم می‌کند .این فرآیند سازگاری به پیش بینی‌کننده‌ی Agree کمک می‌کند تا عملکرد خود را در طول زمان بهینه کند.

به طور کلی، پیش بینی‌کننده‌ی Agree با استفاده از توافق بین مجموعه‌ای از پیش بینی‌کننده‌های مجزا ، پیش بینی‌های دقیق‌تری در مورد نتایج پرش‌ها انجام می‌دهد .با در نظر گرفتن بینش‌های جمعی استراتژی‌های پیش بینی متنوع، می‌تواند به طور مؤثر در سناریوهای مختلف اجرای برنامه حرکت کند و دقت کلی پیش بینی را بهبود بخشد.

سوالی که به وجود می آید چرا ما از بقیه شبکه های عصبی برای پیش بینی استفاده نمی کنیم ؟ چرا فقط پرسپترون ؟

پیچیدگی و سربار :شبکه های عصبی، به ویژه شبکه های عصبی عمیق، می توانند از نظر محاسباتی سنگین و از نظر حافظه پرمصرف باشند .پیاده سازی آنها در سخت افزار برای پیش بینی پرش می تواند پیچیدگی و سربار پردازنده را به طور قابل توجهی افزایش دهد که به طور بالقوه هر گونه افزایش دقت پیش بینی را جبران می کند.

سربار آموزش :آموزش یک شبکه عصبی برای پیش بینی پرش به مجموعه داده بزرگی از تاریخچه و نتایج پرش نیاز دارد .در حالی که چنین داده هایی را می توان از طریق شبیه سازی یا اجرای واقعی جمع آوری کرد، فرآیند آموزش به خودی خود می تواند زمان بر و پرمصرف باشد.

تأخیر :شبکه های عصبی به طور معمول به دلیل محاسبات مورد نیاز برای استنتاج، تأخیر بیشتری ایجاد می کنند .در زمینه پیش بینی پرش، جایی که دقت و سرعت پیش بینی برای حفظ عملکرد بالای پردازنده حیاتی است، هرگونه تأخیر اضافی معرفی شده توسط پیش بینی کننده های مبتنی بر شبکه عصبی می تواند مضر باشد.

پویایی :رفتار الگوهای پرش در برنامه ها می تواند بر اساس عوامل مختلفی مانند داده های ورودی، فازهای برنامه و شرایط اجرا به صورت پویا تغییر کند .شبکه های عصبی ممکن است در مقایسه با سایر پیش بینی کننده هایی که به طور خاص برای مدیریت کارآمد رفتار پویا طراحی شده اند، برای سازگاری سریع با چنین تغییراتی دچار مشکل شوند.

مصرف انرژی :پیاده سازی شبکه های عصبی در سخت افزار می تواند مصرف انرژی را افزایش دهد که یک ملاحظه مهم در طراحی پردازنده های مدرن، به ویژه در دستگاه های تلفن همراه و مبتنی بر باتری است.

قابل تفسیر بودن :شبکه های عصبی اغلب به عنوان مدل های جعبه سیاه شناخته می شوند، به این معنی که درک اینکه چرا آنها پیش بینی های خاصی را انجام می دهند، می تواند چالش برانگیز باشد .در مقابل، پیش بینی کننده های پرش ای سنتی مانند پیش بینی کننده های تورنمنت شفافیت بیشتری ارائه می دهند و تحلیل و بهینه سازی رفتار آنها را برای معماران آسان تر می کنند.