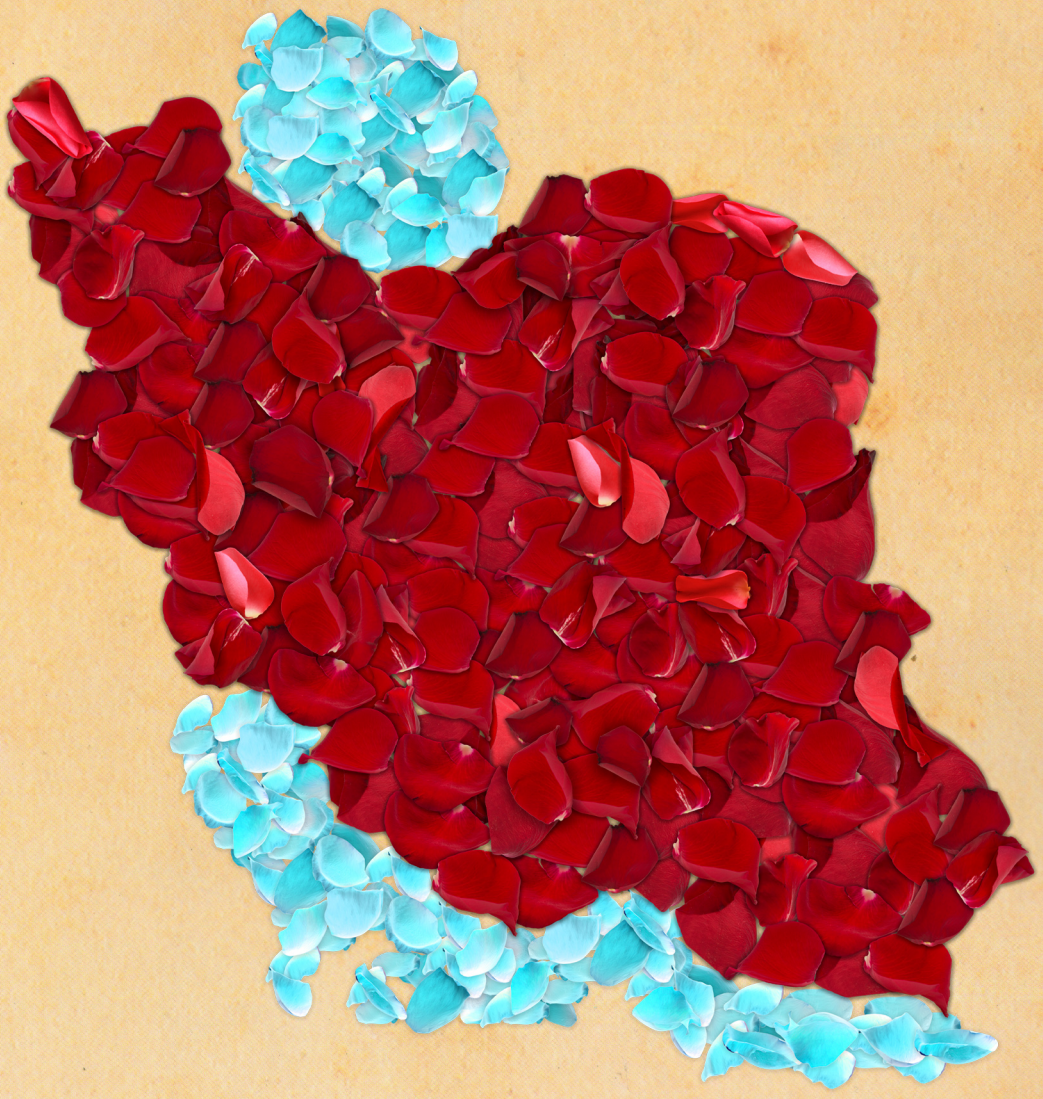
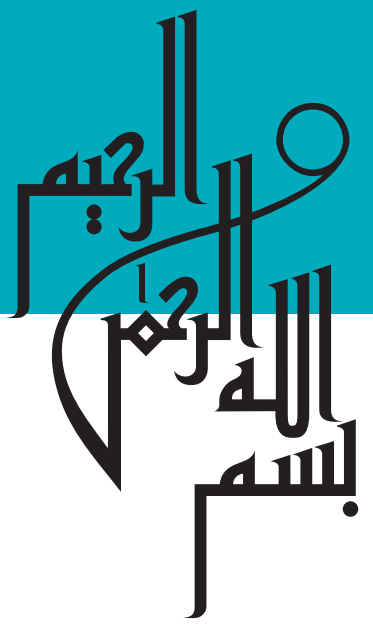
ماتریس

۶

صاحب امتیاز : انجمن علمی مهندسی کامپیوتر
دانشگاه شاهد | خرداد ماه ۱۴۰۴



در این شماره میخوانیم :


بسم الله الرحمن الرحيم

شناسنامه

نشریه ماتریس

صاحب امتیاز: انجمن علمی مهندسی کامپیوتر دانشگاه شاهد

مدیر مسئول: علی بقائی راوری

سر دبیر: فربد فیضی

تیم تحریریه این شماره: سارا کاظم زاده عطار | مانلی فروتن | سارا امیرحسینی | امیرحسین ملکی | فاطمه غلامی | علی کاظم پور | محمدمهدی بابابیک

تیم ویراست این شماره: سارا کاظم زاده عطار

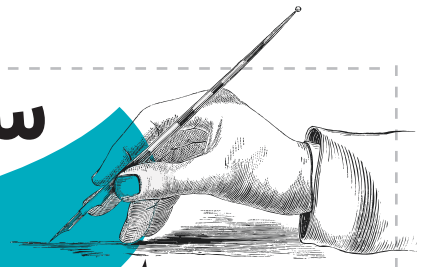
طراح جلد: محمدرضا ناحی داریانی

طراح مجله: سارا جودکی

شبکه‌های اجتماعی: @MatrisMagazine

شماره ششم | خرداد ماه ۱۴۰۴

فهرست



سخن مدیر مسئول

به نام آن‌که دل در گروِ وطن نهاد

گاهی زخم‌ها، بیش از هر کلمه‌ای، انسان‌ها را به هم نزدیک می‌کنند. روزهایی که صدای تهدید در گوشه‌وکنار این سرزمین پیچید، چیزی در دل جامعه‌ی دانشگاهی ایران بیدار شد. نیازی عمیق به معنا، به هم‌صدایی، و به ساختن در برابر ویرانی.

در میان تردیدها و اضطراب‌ها، صدای دانشجو بار دیگر بلند شد؛ نه با هیاهو، که با اندیشه. نه با هیجان گذرا، که با تلاشی جمعی برای آن‌که بگوییم: ما هنوز هستیم. هنوز می‌نویسیم، می‌اندیشیم، و دست‌دردست هم، آینده‌ای بهتر را مجسم می‌کنیم.

ماتریس ششم، نه فقط نتیجه‌ی کار یک تیم، که بخشی از تپش همگانی دانشجویانی‌ست که باور دارند نوشتن، روشن‌ترین شکلِ ایستادگی‌ست.

علی بقائی راوری مدیر مسئول

از جدول تا گره؛ پایگاه داده گرافی

سارا کاظم زاده عطار
مانلی فروتن

چرا گراف دیتابیس به وجود آمد؟

پیش از ظهور پایگاه‌داده‌های گرافی، بیشتر سیستم‌ها از پایگاه های داده رابطه‌ای (Relational Databases) استفاده می‌کردند. در این نوع پایگاه‌داده، اطلاعات در قالب جدول‌هایی از سطر و ستون ذخیره می‌شود و روابط بین داده‌ها معمولاً با استفاده از کلیدهای خارجی و عملیات‌هایی به‌نام JOIN برقرار می‌شود. این ساختار برای بسیاری از کاربردهای سنتی مانند مدیریت اطلاعات کارمندان، کالاها یا سفارش‌ها مناسب و مؤثر بود.

اما با گسترش اینترنت، شبکه‌های اجتماعی، سیستم‌های توصیه‌گر (مثل پیشنهاد فیلم در فیلیمو یا کالا در دیجی‌کالا) و شبکه‌های پیچیده‌تری مانند ساختار وب یا تعاملات زیستی، محدودیت‌های پایگاه های داده رابطه‌ای آشکار شد. تصور کنید بخواهید در یک شبکه اجتماعی، اطلاعات مربوط به «دوستانِ دوستانِ دوستان یک کاربر» را استخراج کنید؛ این کار با پایگاه‌داده رابطه‌ای نیازمند چندین عملیات JOIN سنگین و زمان‌بر است.

از اوایل دهه ۱۳۸۰ (۲۰۰۰ میلادی) تحلیل‌گران متوجه شدند که داده‌ها در بسیاری از موارد به‌صورت شبکه‌ای یا گرافی هستند:

- کاربران با یکدیگر ارتباط دارند؛
- محصولات به یکدیگر وابسته‌اند؛
- صفحات وب از طریق لینک‌ها به هم وصل‌اند؛
- ژن‌ها و پروتئین‌ها شبکه‌ای پیچیده از تعامل دارند.

در چنین شرایطی، نیاز به مدلی احساس شد که خودِ روابط را به‌عنوان داده‌ی اصلی در نظر بگیرد، نه فقط به‌صورت پیوندی میان جدول‌ها. اینجا بود که مفهوم پایگاه‌داده گرافی (Graph Database) مطرح شد.

در گراف دیتابیس، داده‌ها به‌صورت گره (Node) و لبه (Edge) ذخیره می‌شوند - درست مانند نظریه گراف در ریاضیات. این مدل باعث می‌شود تا بررسی روابط پیچیده بین داده‌ها سریع‌تر، ساده‌تر و طبیعی‌تر انجام شود.

البته گراف دیتابیس ها جایگزین کامل سایر پایگاه داده ها نیستند؛ بلکه انتخاب نوع پایگاه داده به نیاز پروژه، نوع داده‌ها و الگوی استفاده بستگی دارد. در بسیاری از موارد، ترکیب چند نوع پایگاه داده می‌تواند بهترین نتیجه را بدهد.

در مقایسه با پایگاه های داده رابطه‌ای

گراف دیتابیس‌ها در تحلیل داده‌های پیچیده و به‌شدت مرتبط مانند شبکه‌های اجتماعی یا زنجیره تأمین عملکرد بهتری دارند، اما برای تراکنش‌های مالی پیچیده مانند سیستم‌های بانکی سنتی، پایگاه‌داده‌های رابطه‌ای همچنان انتخاب اصلی هستند.

در کنار دیگر NoSQLها

در بسیاری از پروژه‌ها، گراف دیتابیس در کنار پایگاه‌های سندمحور مثل MongoDB استفاده می‌شود؛ مثلاً برای نمایش ساختار سلسله‌مراتبی داده‌ها و در عین حال تحلیل روابط آن‌ها.

کاربردهای گراف دیتابیس:

۱. سیستم‌های توصیه‌گر (Recommendation Sys-tems)

در پلتفرم‌هایی مانند دیجی‌کالا یا نماوا، کاربران هزاران محصول یا محتوا را مشاهده، جست‌وجو یا خرید می‌کنند. سیستم‌های توصیه‌گر تلاش می‌کنند بر اساس این رفتارها، محصولات مرتبط را به کاربران پیشنهاد دهند. اینجا با استفاده از گراف دیتابیس می توان ارتباط بین کاربران و کالاها را در قالب گره و رابطه نمایش داد.

مثال: فرض کنید کاربر A و کاربر B هر دو گوشی و هندزفری مشابهی خریده‌اند. حالا اگر کاربر B یک شارژر خاص هم خریده باشد، سیستم می‌تواند آن را به کاربر A پیشنهاد دهد، چون این دو کاربر الگوی خرید مشابهی دارند.

۲. کشف تقلب (Fraud Detection)

در سیستم‌های بانکی و مالی، تشخیص تراکنش‌های مشکوک اهمیت زیادی دارد. تقلب‌ها معمولاً به‌صورت الگوهای پیچیده و غیرمستقیم رخ می‌دهند، نه به‌صورت مستقیم. در اینجا گراف دیتابیس کمک می‌کند تا ارتباط‌های غیرعادی میان حساب‌ها، کارت‌ها یا آدرس‌های IP شناسایی شود. مثلا اگر چند حساب بانکی مختلف که به ظاهر بی‌ربط هستند، همگی در بازه زمانی کوتاهی از یک دستگاه مشترک تراکنش انجام دهند، یا وجوهی بین آن‌ها با الگوی غیرعادی منتقل شود، می‌توان این موضوع را به‌عنوان تقلب احتمالی در نظر گرفت.

۳. مدیریت دانش (Knowledge Management)

در مراکز پژوهشی و دانشگاه‌ها، ارتباط میان پژوهشگران، مقالات، موضوعات و واژه‌های کلیدی، یک شبکه‌ی بزرگ علمی می‌سازد. گراف دیتابیس می‌تواند این روابط را نمایش دهد و تحلیل‌هایی مانند یافتن نویسندگان همکار، مقالات مشابه، یا حوزه‌های پژوهشی نوظهور را آسان‌تر کند.

برای مثال یک پژوهشگر ممکن است بخواهد بداند که «چه کسانی در زمینه یادگیری ماشین و پردازش زبان طبیعی، بیشترین همکاری را داشته‌اند؟» یا «چه موضوعاتی در مقالات منتشرشده درباره هوش

مصنوعی در حال رشد هستند؟»

٤. تحلیل شبکه‌های اجتماعی (Social Network Analysis)

در شبکه‌هایی مانند اینستاگرام، لینکدین یا توییتر، ارتباط بین کاربران (دنبال کردن، لایک کردن، پیام دادن، تگ کردن و...) شبکه‌ای پیچیده از تعاملات را شکل می‌دهد. گراف دیتابیس‌ها می‌توانند برای تحلیل این شبکه‌ها استفاده شوند و اطلاعات ارزشمندی درباره ساختار و رفتار کاربران ارائه دهند. از جمله:

- شناسایی کاربران تأثیرگذار (Influencers) که بیشترین تأثیر را روی سایر کاربران دارند
- یافتن گروه‌های پنهان یا بسته‌هایی از افراد که تعامل زیادی با هم دارند، ولی با بقیه شبکه ارتباطی ندارند (Community Detection)
- بررسی مسیر انتشار یک خبر یا شایعه در میان کاربران (Propagation Tracking)

این کاربردها در بازاریابی، سیاست، امنیت اجتماعی و حتی مطالعات فرهنگی کاربرد زیادی دارند.

اما با وجود مزایای گراف دیتابیس، پیاده‌سازی آن در مقیاس وسیع با چالش‌هایی همراه است. برخی از این چالش‌ها و راه‌حل‌های رایج آن‌ها عبارت‌اند از:

- مقیاس‌پذیری

وقتی با گراف‌هایی سروکار داریم که شامل میلیاردها گره و رابطه هستند، نگهداری و پردازش آن‌ها کار آسانی نیست .

راهکار : استفاده از معماری‌های توزیع‌شده مانند Neo4j Fabric یا تکنیک‌های پارتیشن‌بندی هوشمند برای تقسیم گراف به بخش‌های قابل مدیریت .

- بهینه‌سازی پرس‌وجو

جستجوهایی مانند «دوستِ دوستِ دوستِ...» ممکن است کند و زمان‌بر باشند.

راهکار: بهره‌گیری از الگوریتم‌های هوشمند و شاخص‌گذاری بر اساس نوع روابط.

- سازگاری داده‌ها

در همه‌ی سیستم‌ها نیازی نیست که داده‌ها همیشه کاملاً دقیق و بلافاصله به‌روز باشند. بسته به نوع کاربرد، می‌توان بین مدل‌های مختلف سازگاری داده انتخاب کرد.

مدل ACID برای سیستم‌های حساس مانند بانکداری که به دقت و صحت بالا نیاز دارند مناسب است،

در حالی که مدل BASE برای تحلیل‌های سریع و گسترده، مانند شبکه‌های اجتماعی یا سیستم‌های توصیه‌گر، عملکرد بهتری دارد.

راهکار: انتخاب بین مدل‌های ACID و BASE باید با توجه به نیاز پروژه انجام شود.

در دنیایی که همه‌چیز به‌نوعی به هم مرتبط است، گراف دیتابیس‌ها فقط یک ابزار ذخیره‌سازی نیستند؛ بلکه راهی نو برای تحلیل، فهم، و کشف روابط پنهان در دنیای پیچیده‌ی داده‌ها هستند.

سامانه Rocks Cluster

معماری یکپارچه برای ساخت و مدیریت ابررایانه‌ها

سارا امیرحسینی
امیرحسین ملکی

بحران پیچیدگی در محاسبات با کارایی بالا (HPC)

در عصر داده‌های بزرگ، نیاز به توان پردازشی عظیم در حوزه‌هایی مانند فیزیک، بیوانفورماتیک، هوش مصنوعی و مهندسی به یک ضرورت تبدیل شده است. رایانش خوشه‌ای (Cluster Computing) با اتصال ده‌ها یا صدها رایانه معمولی (Commodity Hardware) از طریق شبکه‌های پرسرعت، راهکاری اقتصادی برای ایجاد ابررایانه‌های مجازی ارائه می‌دهد. اما این سادگی مفهومی با چالش‌های عملی بزرگی همراه است: نصب سیستم‌عامل، پیکربندی شبکه، مدیریت کاربران، نصب نرم‌افزارهای علمی، نظارت بر گره‌ها و تضمین پایداری سیستم، فرآیندی پیچیده، زمان‌بر و مستعد خطای انسانی است.

سوال کلیدی این بود: چگونه می‌توان مدیریت خوشه‌های محاسباتی را از یک «هنر دستی» به یک «فرآیند صنعتی، خودکار و تکرارپذیر» تبدیل کرد؟ پاسخ این سوال، پروژه‌ای به نام Rocks Cluster Distribution بود که در سال ۲۰۰۰ در مرکز سوپرکامپیوتر سن‌دیگو (SDSC) در دانشگاه کالیفرنیا متولد شد.

Rocks Cluster چیست؟

Rocks Cluster Distribution (یا به اختصار Rocks) یک توزیع لینوکسی متن‌باز است که به‌طور خاص برای ساخت، پیکربندی و مدیریت خوشه‌های محاسباتی با کارایی بالا (HPC) طراحی شده است. این سامانه، مبتنی بر توزیع‌های معتبر لینوکس مانند CentOS (و در نسخه‌های جدیدتر Rocky Linux)، یک راه‌حل یکپارچه ارائه می‌دهد که تمام جنبه‌های چرخه حیات یک خوشه را پوشش می‌دهد: از نصب اولیه تا مدیریت روزمره و مقیاس‌پذیری.

Rocks با شعار «سادگی در عین قدرت» امکان راه‌اندازی خوشه‌های چند صد گره‌ای را در چند ساعت فراهم می‌کند، در حالی که در روش‌های سنتی این فرآیند ممکن است هفته‌ها طول بکشد. این سامانه به‌ویژه در محیط‌های دانشگاهی، آزمایشگاهی و تحقیقاتی که منابع انسانی و مالی محدودی دارند، محبوب است.

فلسفه Rocks: حل مشکل «ضعیف‌ترین حلقه زنجیره»

توسعه‌دهندگان Rocks با یک مشاهده کلیدی کار خود را آغاز کردند:

«ضعیف‌ترین حلقه در پایداری یک خوشه محاسباتی، مدیر سیستم (HPC Admin) است.»

مدیران سیستم، با وجود مهارت‌هایشان، به دلیل تغییرات دستی و غیرمستند (مثل نصب یک بسته نرم‌افزاری، تغییر فایل پیکربندی یا اجرای اسکریپت‌های سفارشی) می‌توانند خوشه را از حالت پایدار خارج کنند. این تغییرات، خوشه را به سیستمی شکننده و غیرقابل تکرار تبدیل می‌کنند که عیب‌یابی آن در صورت بروز مشکل تقریباً غیرممکن است.

Rocks با دو اصل بنیادی به این چالش پاسخ داد:

۱.خودمختاری و بازتولیدپذیری (Automation & Reproducibility): تمام فرآیندهای نصب، به‌روزرسانی و پیکربندی به‌صورت خودکار از طریق گره مرکزی (Frontend) انجام می‌شود. مدیر سیستم به‌ندرت نیاز به دخالت مستقیم در گره‌های محاسباتی دارد. اگر گره‌ای دچار مشکل شود، با یک راه‌اندازی مجدد، سیستم‌عامل و پیکربندی آن به‌صورت خودکار از Frontend بازنصب می‌شود.

۲. خوشه به‌مثابه یک دستگاه واحد (Cluster as an Appliance) خوشه را به‌عنوان یک سیستم یکپارچه می‌بیند، نه مجموعه‌ای از رایانه‌های مستقل. این رویکرد، مدیریت را از سطح گره‌های منفرد به سطح کل خوشه منتقل می‌کند.

این فلسفه، مدیریت خوشه را از یک فرآیند پرخطا به یک علم مهندسی دقیق تبدیل کرد و پایداری و پیش‌بینی‌پذیری را تضمین نمود.

معماری و اجزای کلیدی Rocks

معماری Rocks بر پایه یک ساختار مدولار و توزیع‌شده طراحی شده است که شامل اجزای زیر است:

۱. گره مرکزی (Front-end Node)

گره مرکزی یا Front-end، مغز خوشه است و تمام سرویس‌های مدیریتی را اجرا می‌کند:

- سرور نصب (Installation Server): با استفاده از PXE (Preboot Execution Environment) و Kickstart، سیستم‌عامل را به‌صورت خودکار روی گره‌های محاسباتی نصب می‌کند.
- سرویس‌های زیرساختی: شامل DHCP، DNS، NTP و NFS برای مدیریت شبکه و همگام‌سازی.
- مخزن نرم‌افزاری: بسته‌های نرم‌افزاری و پیکربندی‌ها از این گره به سایر گره‌ها توزیع می شوند .
- نقطه ورود مدیر: تمام تعاملات مدیر سیستم از طریق این گره انجام می شود .

۲. گره‌های محاسباتی (Compute Nodes)

این گره‌ها، نیروی پردازشی خوشه را تأمین می‌کنند و به‌صورت بدون حالت (Stateless) طراحی شده‌اند. آن‌ها هیچ اطلاعات دائمی روی دیسک محلی ذخیره نمی‌کنند و سیستم‌عامل و نرم‌افزارهایشان را از Frontend دریافت می‌کنند. این ویژگی، گره‌ها را قابل تعویض و مقاوم در برابر خرابی می‌کند.

۳. گره‌های خدماتی و ذخیره‌سازی

- گره‌های ذخیره‌سازی (Storage Nodes): برای مدیریت داده‌های بزرگ و فایل‌سیستم‌های توزیع‌شده مانند Lustre یا GlusterFS.
- گره‌های خدماتی (Service Nodes): برای اجرای سرویس‌های خاص مانند پایگاه‌داده یا وب‌سرور.

٤. بسته‌های نرم‌افزاری (Rolls)

Rolls، بسته‌های ماژولار نرم‌افزاری هستند که قابلیت‌های خاصی به خوشه اضافه می‌کنند. هر Roll شامل فایل‌های RPM، اسکریپت‌های پیکربندی و تعاریف XML است.

نمونه‌های کلیدی:

- Base Roll: هسته اصلی سیستم Rocks.
- HPC Roll: ابزارهای محاسباتی مانند MPI، OpenMPI و کامپایلرهای GCC و Intel.
- SGE/SLURM Roll: سیستم‌های مدیریت صف مانند Sun Grid Engine و SLURM.

- Bio Roll: ابزارهای بیوانفورماتیک مانند BLAST و Bowtie.
- CUDA/GPU Roll: پشتیبانی از پردازنده‌های گرافیکی NVIDIA.
- Conda Roll: مدیریت محیط‌های پایتون و نرم‌افزارهای علمی.

۵. ابزارهای خودکارسازی

Rocks از ابزارهایی مانند Kickstart، Anaconda و پایگاه‌داده MySQL برای نصب خودکار و مدیریت اطلاعات گره‌ها استفاده می‌کند. این ابزارها فرآیندهای پیچیده را به اسکریپت‌های استاندارد و قابل تکرار تبدیل می‌کنند.

نمادشناسی پروژه: از مار تا صخره

هویت بصری و هستی‌شناسی Rocks ریشه در فرهنگ و فلسفه پروژه دارد:

مار: نماد باززایی و انعطاف‌پذیری

در نسخه‌های اولیه، نماد غیررسمی Rocks یک مار زنگ‌دار (Diamondback Rattlesnake) بود که در کالیفرنیای جنوبی فراوان است. این انتخاب به دلایل زیر انجام شد:

- پوست‌اندازی و تجدید: مارها با پوست‌اندازی، خود را بازسازی می‌کنند. در Rocks، گره‌های محاسباتی با هر نصب مجدد، پیکربندی معیوب را کنار گذاشته و به حالت پایدار بازمی‌گردند.
- یکپارچگی و قدرت: مار به‌عنوان موجودی واحد و منسجم، نماد خوشه‌ای است که به‌صورت یک دستگاه یکپارچه عمل می‌کند.
- ارتباط فرهنگی: مار در فرهنگ بومیان آمریکا نماد استقامت و قدرت است و با محیط کالیفرنیا همخوانی دارد.

صخره: نماد پایداری و استحکام

با رسمی‌تر شدن پروژه، لوگو به یک صخره تغییر یافت که نشان‌دهنده:

- ثبات و پایداری: خوشه‌های Rocks برای محاسبات طولانی‌مدت و حیاتی طراحی شده‌اند.
- بنیان محکم: Rocks زیرساختی قابل اعتماد برای تحقیقات علمی فراهم می‌کند.

مزایا و معایب Rocks Cluster

مزایا

۱. نصب سریع و خودکار: راه‌اندازی خوشه در چند ساعت به جای هفته‌ها.

۲. مدیریت متمرکز: تمام عملیات از طریق Frontend انجام می‌شود.

۳. تکرارپذیری بالا: پیکربندی یکسان در تمام گره‌ها و امکان بازسازی سریع.

٤. مقیاس‌پذیری عالی: از خوشه‌های کوچک تا سیستم‌های چند هزار گره‌ای.

۵. اکوسیستم غنی Rolls: پشتیبانی از کاربردهای متنوع از بیوانفورماتیک تا هوش مصنوعی.

٦. پشتیبانی از سخت‌افزار متنوع: از سیستم‌های قدیمی تا سرورهای مدرن با GPU.

۷. جامعه کاربری فعال: مستندات گسترده و انجمن‌های آنلاین.

معایب

۱. عدم انعطاف‌پذیری در پیکربندی‌های خاص: رویکرد خودکار برای مدیرانی که به کنترل دستی عادت دارند، محدودکننده است.

۲. کاهش سرعت توسعه: از سال ۲۰۱۵، به‌روزرسانی‌ها کند شده و Rocks در برابر فناوری‌های جدیدتر مانند Kubernetes عقب مانده است.

۳. نقطه تکی شکست: خرابی Frontend می‌تواند مدیریت خوشه را مختل کند (هرچند محاسبات در حال اجرا ادامه می‌یابند).

٤. وابستگی به لینوکس: کاربران غیرآشنا با لینوکس ممکن است با چالش‌هایی مواجه شوند.

۵. پیچیدگی در مقیاس‌های بزرگ: مدیریت خوشه‌های بسیار بزرگ نیاز به دانش تخصصی دارد.

چالش‌های مدیریت خوشه‌های HPC و راه‌حل Rocks

مدیریت خوشه‌های HPC با چالش‌های متعددی مواجه است:

- تغییرات مکرر نرم‌افزارها: به‌روزرسانی دستی نرم‌افزارها زمان‌بر و مستعد خطاست.
- مدیریت منابع: تخصیص منابع به کاربران و مدیریت صف‌های کاری.
- پیکربندی شبکه: تضمین ارتباط سریع و پایدار بین گره‌ها.
- امنیت و دسترسی: مدیریت حساب‌های کاربری و جلوگیری از نفوذ.
- Rocks این چالش‌ها را با رویکردهای زیر حل کرده است:
- خودکارسازی کامل: نصب و به‌روزرسانی از طریق Kickstart و اسکریپت‌های استاندارد.
- مدیریت متمرکز: Frontend تمام گره‌ها را هماهنگ می‌کند.
- سیستم‌های صف‌بندی: Rollsهایی مانند SLURM وSGE وظایف را بهینه مدیریت می‌کنند.
- پایگاه‌داده متمرکز: ذخیره اطلاعات گره‌ها و کاربران در MySQL.

این خودکارسازی، نقش مدیر سیستم را از «ضعیف‌ترین حلقه» به یک ناظر ساده تبدیل کرده است.

تاریخچه و تکامل Rocks

خواستگاه

پروژه Rocks در سال ۲۰۰۰ توسط تیمی به رهبری فیلیپ پاپادوپولوس، گرگ برونو و مایکل کتز در SDSC آغاز شد. هدف اولیه، ساده‌سازی مدیریت خوشه‌های تحقیقاتی بود. مقاله اصلی پروژه در کنفرانس IEEE Cluster ۲۰۰۱ منتشر شد و Rocks را به‌عنوان یک نوآوری کلیدی معرفی کرد.

تکامل

- ۲۰۰۰-۲۰۰۵: انتشار نسخه‌های اولیه با تمرکز بر نصب خودکار و پشتیبانی از MPI.
- ۲۰۰۵-۲۰۱۰: معرفی Rollsهای جدید مانند Bio Roll و SGE Roll.
- ۲۰۱۰-۲۰۱۵: ادغام با GPUها و سیستم‌های ذخیره‌سازی توزیع‌شده.
- ۲۰۱۵ به بعد: کاهش سرعت توسعه به دلیل ظهور Kubernetes و OpenHPC، اما استفاده مداوم در محیط‌های آموزشی.

کاربردها و پروژه‌های اجرا شده

Rocks در حوزه‌های متعددی استفاده شده است:

۱. محاسبات علمی:

- شبیه‌سازی‌های عددی در فیزیک (دینامیک سیالات محاسباتی).
- مدل‌سازی نجومی برای تحلیل داده‌های تلسکوپ ها .
- محاسبات شیمی کوانتومی با ابزارهایی مانند Gaussian و NWChem.

۲. بیوانفورماتیک:

- تحلیل داده‌های ژنومی با BLAST و Bowtie.
- مدل‌سازی پروتئین‌ها با Rosetta.

۳. هوش مصنوعی و یادگیری ماشین:

- آموزش مدل‌های یادگیری عمیق با TensorFlow و PyTorch.
- پردازش کلان‌داده‌ها.

٤. پروژه‌های دانشگاهی:

- دانشگاه‌های MIT، Stanford، UCSD، ETH Zurich و دانشگاه‌های ایرانی مانند شریف و امیرکبیر از Rocks در شبیه‌سازی‌های مهندسی و تحلیل داده‌های زیستی استفاده کرده‌اند.

آینده Rocks Cluster

با وجود کاهش سرعت توسعه، Rocks همچنان در محیط‌های آموزشی و تحقیقاتی کوچک کاربرد دارد. روندهای آینده شامل:

۱. ادغام با فناوری‌های مدرن: استفاده از Docker و Singularity برای اجرای برنامه‌های جدیدتر.

۲. رقابت با OpenHPC و Kubernetes: سادگی Rocks برای پروژه‌های کوچک همچنان مزیت دارد.

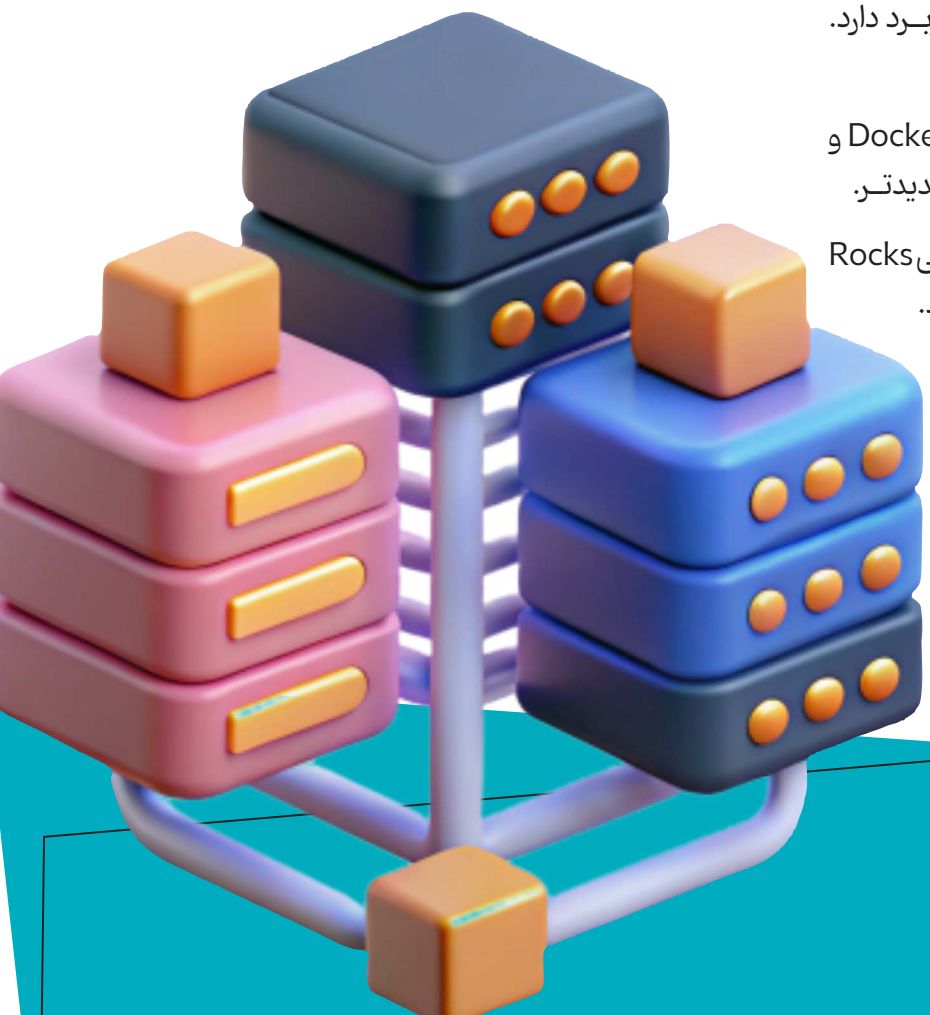
۳. نقش آموزشی: ابزاری برای یادگیری مفاهیم رایانش خوشه‌ای.

٤. محاسبات ابری: ادغام با AWS یا Azure برای خوشه‌های هیبریدی.

نتیجه‌گیری

Rocks Cluster Distribution فراتر از یک نرم‌افزار، یک فلسفه در مدیریت سیستم‌های پیچیده است. با خودکارسازی فرآیندها، کاهش وابستگی به مدیر سیستم و ارائه یک اکوسیستم ماژولار، Rocks راه را برای ساخت خوشه‌های پایدار و مقیاس‌پذیر هموار کرد. هرچند فناوری‌های جدیدتر در حال جایگزینی هستند، میراث Rocks در مفاهیمی مانند زیرساخت به‌مثابه کد و مدیریت اعلانی همچنان زنده است. برای

دانشجویان و پژوهشگران، Rocks نه‌تنها یک ابزار، بلکه درسی عمیق در اهمیت اتوماسیون و بازتولیدپذیری در علم و مهندسی است.



دنیای کوانتومی ۲ ؛
آینده‌ای که داره ساخته میشه!

فاطمه غلامی

محاسبات کلاسیک در برابر محاسبات کوانتومی چه تفاوتی دارند؟

در کامپیوترهای کلاسیک، هر برنامه مثل یه مدار منطقی ساخته می‌شه؛ یعنی اطلاعات (۰ و ۱) از گیت‌های منطقی عبور می‌کنن و در نهایت به جواب می‌رسیم. این مدل، ساده و قابل فهمه ولی برای بعضی مسائل پیچیده مثل فاکتورگیری اعداد خیلی بزرگ یا جستجوی سریع تو پایگاه داده‌های عظیم، خیلی کند و وقت‌گیر می‌شه.

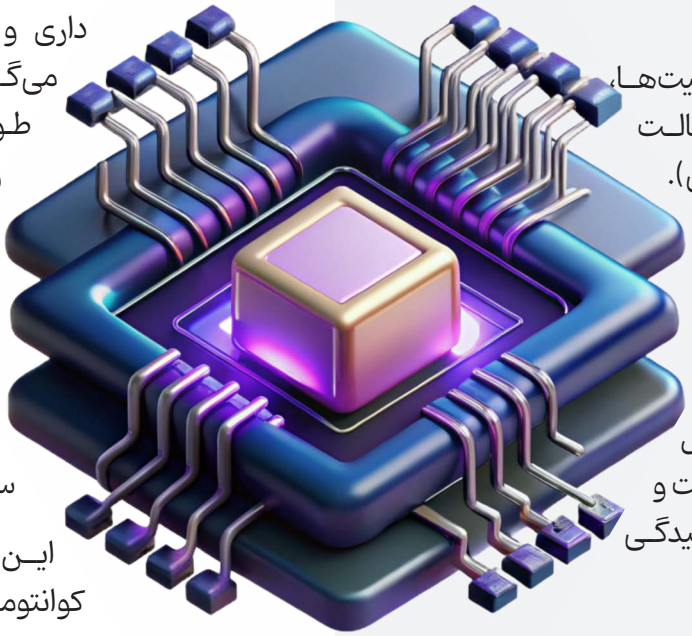
مدل مدار کوانتومی چیه؟

محاسبات کوانتومی بر پایه کیوبیت‌ها و گیت‌های کوانتومی ساخته می‌شه؛ مشابه مدارهای کلاسیک ولی با تفاوت‌های اساسی که قبلتر گفتیم؛

کیوبیت‌ها به جای بیت‌ها، می‌تونن همزمان چند حالت رو داشته باشن (برهم‌نهی).

گیت‌های کوانتومی مثل Hadamard، Pauli و CNOT، به صورت واحدهای منطقی کوانتومی عمل می‌کنن و قادر به ساخت و کنترل برهم‌نهی و درهم‌تنیدگی هستن .

مدار کوانتومی، دنباله‌ای از این گیت‌هاست که روی کیوبیت‌ها اعمال می‌شه تا محاسبات رو انجام بده.



الگوریتم Shor و Grover انقلابی تو محاسبات

- **الگوریتم Shor برای فاکتورگیری اعداد بزرگ**

در کامپیوترهای کلاسیک، فاکتورگیری اعداد بزرگ به اعداد اول (مثلاً شکستن اعداد بزرگ به عوامل اول) به شدت سخت و زمان‌بره، مخصوصاً وقتی عدد خیلی بزرگ باشه. این سختی پایه امنیت بسیاری **از** سیستم‌های رمزنگاریه.

اما الگوریتم Shor که در سال ۱۹۹۴ توسط Peter Shor ارائه شد، می‌تونه این کار رو به صورت نمایی سریع‌تر انجام بده! یعنی به جای اینکه هزاران سال طول بکشه، با یک کامپیوتر کوانتومی مناسب در چند دقیقه یا ساعت انجام می‌شه.

این الگوریتم از مدل مدار کوانتومی استفاده می‌کنه و ترکیبی از برهم‌نهی و درهم‌تنیدگی رو به کار می‌بره تا سریع‌ترین مسیر برای فاکتورگیری رو پیدا کنه.

- **الگوریتم Grover (برای جستجوی سریع)**

فرض کن یک پایگاه داده خیلی بزرگ داری و دنبال یک ورودی خاص می‌گردی. کامپیوتر کلاسیک به طور متوسط باید نصف داده‌ها رو جستجو کنه تا جواب رو پیدا کنه (عملکرد خطی).

الگوریتم Grover اما این کار رو تقریبا در ریشه دوم زمان ($\sqrt{N}$) انجام می‌ده؛ یعنی خیلی سریع تر !

این الگوریتم هم از گیت‌های کوانتومی برای انجام جستجوی موازی روی حالات مختلف کیوبیت‌ها استفاده می‌کنه.

پس چی شد؟ مدل مدار کوانتومی باعث شد که ما دیگه محدود به محاسبات سریالی و خطی نباشیم. با ترکیب کیوبیت‌ها و گیت‌های کوانتومی و استفاده از الگوریتم‌های خاص، توانایی حل مسائل بسیار پیچیده و حساس به شکل نمایی افزایش پیدا کرده.

چرا محاسبات کوانتومی اینقدر آینده‌داره؟

خب، تا حالا متوجه شدیم چرا کامپیوترهای معمولی دیگه اون سرعت رشد و قدرت قبلی رو ندارن؟ چون قانون مور این روزها انگار خیلی کند شده. اینجاست که محاسبات کوانتومی میاد وسط و می‌گه: «هی بچه‌ها، یه راه بهتر دارم!» همون طور که گفتیم ، محاسبات کوانتومی پتانسیل باور نکردنی در حل مسائل داره .

اما الان تو چه مرحله‌ای هستیم؟

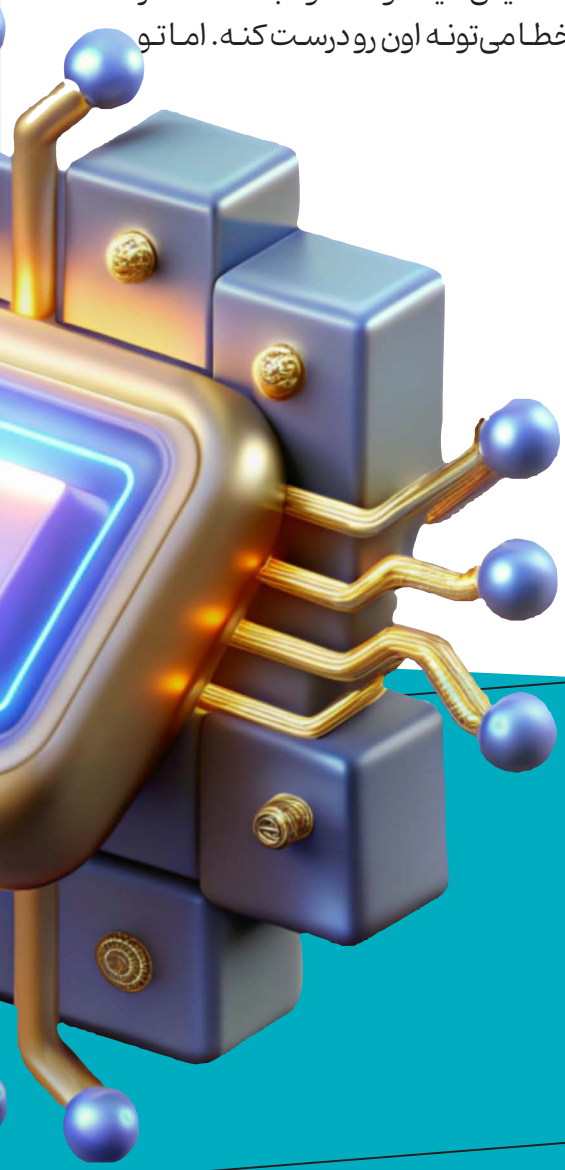
الان ما توی یه مرحله به اسم «NISO» هستیم. این یعنی کامپیوترهای کوانتومی‌ای که ساختیم هنوز کامل نیستن، یه عالمه خطا و نویز دارن و فقط می‌تونن یه سری کارهای محدود انجام بدن. ولی همین کامپیوترهای ناپایدار دارن دنیای محاسبات رو کم‌کم تغییر می‌دن و آماده می‌کنن برای روزی که کامپیوترهای کوانتومی کامل‌تر و قدرتمندتر بیان سر کار.

چرا این مهمه؟ چون حتی همین کامپیوترهای نیمه‌کاره (NISO) هم می‌تونن مسائلی رو حل کنن که کامپیوترهای کلاسیک واقعا خیلی سخت یا حتی غیرممکن بتونن انجام بدن. این یعنی دنیای رمزنگاری، داروسازی، هوش مصنوعی و بهینه‌سازی قراره کلی تغییر کنه و همه اینها دست‌خوش پیشرفت‌های کوانتومی می‌شه.

چالش‌های فنی و علمی محاسبات کوانتومی

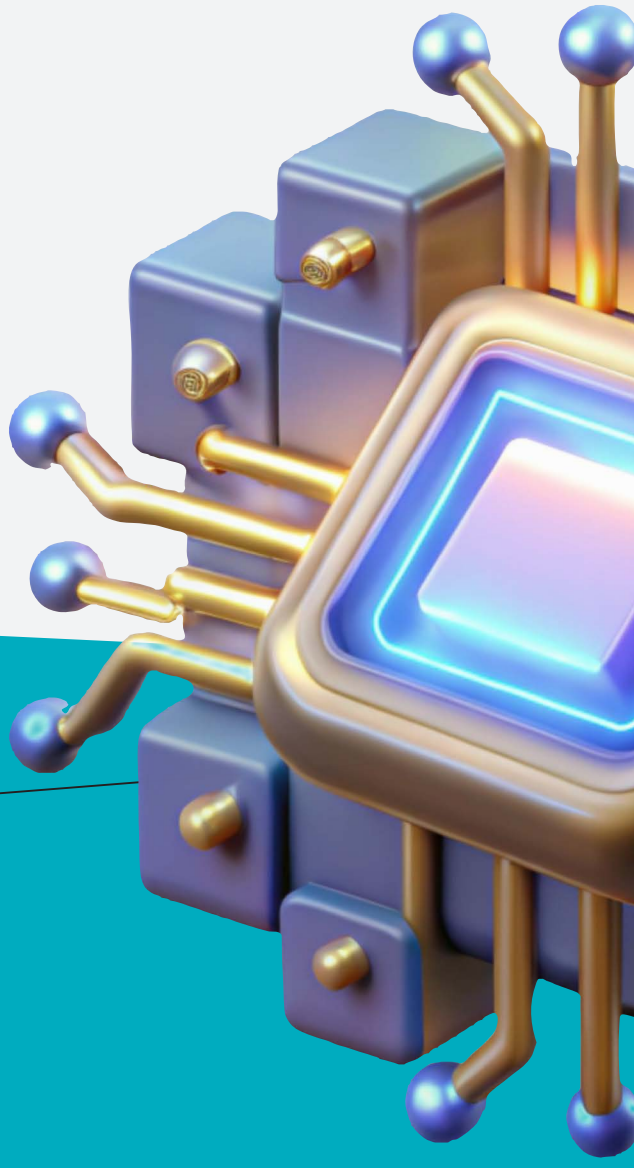
کیوبیت‌ها مثل بیت‌های کامپیوترهای معمولی نیستن که یا صفر باشن یا یک. این ذرات کوانتومی خیلی حساسن و به راحتی تحت تأثیر نویز و عوامل محیطی قرار می‌گیرن. این یعنی هر لحظه ممکنه اطلاعات کوانتومی‌شون خراب بشه یا تغییر کنه. چیزی که توی دنیای کامپیوترهای معمولی خیلی کمتر اتفاق می‌افته. یکی از چالش‌های اصلی اینه که چطور می‌تونیم این خطاها رو کنترل کنیم و کیوبیت‌ها رو پایدار نگه داریم تا محاسبات درست انجام بشه.

در دنیای کامپیوترهای معمولی، اگر یه بیت خراب بشه، سیستم خطا رو تشخیص میده و معمولاً با استفاده از کدهای تصحیح خطا می‌تونه اون رو درست کنه. اما تو



دنیای کوانتومی، چون خود ذره‌ها در حالت برهم‌نهی و درهم‌تنیدگی هستن، نمیشه مثل روش‌های کلاسیک عمل کرد. اینجا باید از روش‌های خاص و پیچیده‌ای مثل «کدهای تصحیح خطای کوانتومی» استفاده کنیم که می‌تونن بدون این که اطلاعات اصلی رو بخونن، خطا رو تشخیص و اصلاح کنن.

کیوبیت‌ها برای درست کار کردن نیاز دارن در دماهای بسیار پایین، نزدیک به صفر مطلق (حدود چند میلی‌کِلوین) نگه داشته بشن. این کار نیازمند تجهیزات خنک‌کننده بسیار پیشرفته و پرهزینه هست. این که چطور این دماهای فوق‌العاده پایین رو حفظ کنیم و در عین حال کیوبیت‌ها رو به تعداد بالا بسازیم، یکی از بزرگ‌ترین موانع فنی پیش روی محاسبات کوانتومی محسوب می‌شه.



علاوه بر سخت‌افزار پیچیده، طراحی الگوریتم‌ها و نرم‌افزارهایی که بتونن از مزیت‌های کوانتومی استفاده کنن هم کار ساده‌ای نیست. باید الگوریتم‌هایی نوآورانه نوشته بشه که بتونن با خطاهای احتمالی کنار بیان و بهترین نتیجه رو ارائه بدن. این یعنی جامعه علمی و مهندسی هنوز در ابتدای راه توسعه ابزارهای نرم‌افزاری کوانتومی هست.

- **وضعیت فعلی جهان در حوزه محاسبات کوانتومی ۲۰۲۵**

وضعیت فناوری و بازار جهانی

طبق گزارش شرکت McKinsey و Ouantum Insider، دنیا الان تو یک دورهٔ هیجان‌انگیز در مسیر توسعه محاسبات کوانتومی قرار داره. شرکت‌های بزرگی مثل IBM، Google، Microsoft، Intel و چند استارتاپ نوپا، هر کدوم دارن برای ساخت کیوبیت‌های بیشتر، پایدارتر و با کیفیت‌تر رقابت می‌کنن.

این رقابت نه فقط تو سخت‌افزار، بلکه در توسعه نرم‌افزار، الگوریتم‌ها و بسترهای ابری کوانتومی هم داغه. IBM با برنامه‌ریزی بلندمدت تا سال ۲۰۳۳، نقشه راه روشنی برای تولید کامپیوترهای کوانتومی بزرگ و در دسترس عموم ترسیم کرده.

برای مثال، IBM Ouantum Roadmap وعده داده تا ظرف چند سال آینده تعداد کیوبیت‌ها رو به صدها، سپس هزارها و نهایتاً ده‌ها هزارتا برسونه و قابلیت‌های عملیاتی مثل تصحیح خطا رو بهتر کنه.

سیستم‌های عمومی و دسترسی

یکی از اتفاقات مهم اینه که برخی از شرکت‌ها مثل IBM و Google، سیستم‌های کوانتومی خودشونو به صورت ابری (Cloud Ouantum Computing) در اختیار پژوهشگران و شرکت‌ها گذاشتن. یعنی شما لازم نیست هزاران دلار هزینه کنی که یه دستگاه بخری، فقط کافیه از راه دور بهش وصل بشی و ازش استفاده کنی! این موضوع باعث شده که تحقیقات و کاربردهای عملی سریع‌تر پیش بره.

چالش‌های رقابتی

با وجود پیشرفت‌های بزرگ، هنوز مسائلی مثل تعداد محدود کیوبیت‌های پایدار، خطاهای محاسباتی و هزینه‌های سرسام‌آور وجود داره که رقابت رو سخت می‌کنه. اما تمام شرکت‌ها روی این چالش‌ها متمرکز شدن و با بودجه‌های هنگفت، تکنولوژی رو هر روز بهتر می‌کنن.

- **آینده با محاسبات کوانتوم !**

حالا که کلی در مورد محاسبات کوانتومی خوندیم و فهمیدیم چرا دنیای فناوری داره سمتش میره، یه نگاهی به آینده بندازیم.

طبق یه گزارش از Boston Consulting Group، دهه آینده قراره یکی از هیجان‌انگیزترین دوره‌ها برای کوانتوم باشه. شرکت‌ها، دانشگاه‌ها و حتی دولت‌ها حسابی دست به دست هم دادن که این فناوری رو به اوج برسونن.

حالا یه چیز مهم: آموزش و تربیت آدمای متخصص تو این زمینه واقعاً کلید موفقیته. اگه ما دانشجوها و مهندسان به موقع شروع نکنیم به یادگیری، خیلی جا می‌مونیم.

خلاصه اینکه آینده روشنه، ولی باید همه با هم همکاری کنیم؛ دانشگاه و صنعت و سیاست‌گذاران. فقط با این همکاری‌ها می‌تونیم از ظرفیت‌های واقعی کوانتوم استفاده کنیم و دنیا رو متحول کنیم.

Veo3؛
جهشی از متن به سینما در عصر هوش مصنوعی چندوجهی

سارا امیرحسینی

در دنیای پرشتاب هوش مصنوعی، هر نسل جدید از مدل‌های مولد، دریچه‌ای نو به سوی خلاقیت و تعامل انسان و ماشین می‌گشاید. در بهار ۲۰۲۵، گوگل با رونمایی از Veo3، نه‌تنها مرزهای تولید محتوای بصری را جابه‌جا کرد، بلکه با یکپارچه‌سازی تصویر، حرکت، صدا و منطق سینمایی، استانداردی تازه در خلق محتوای چندوجهی تعریف کرد. این مدل، فراتر از یک ابزار، گامی مفهومی به سوی آینده‌ای است که در آن داستان‌سرایی دیجیتال بدون نیاز به تجهیزات فیزیکی ممکن می‌شود.

معماری Veo3: هم‌افزایی تصویر، حرکت و صدا

Veo3 با بهره‌گیری از معماری پیشرفته ترانسفورمر چندوجهی، رویکردی یکپارچه به تولید محتوا ارائه می‌دهد. برخلاف مدل‌های پیشین که فریم‌های ویدیو را به‌صورت مجزا تولید می‌کردند،Veo3 از یک مدل نوآورانه به نامSpace-Time-Aware Diffusion استفاده می‌کند. این مدل با درک عمیق از روابط فضایی و زمانی بین فریم‌ها، ویدیوهایی روان و منسجم خلق می‌کند که حس و حال یک اثر سینمایی واقعی را منتقل می‌کنند.

علاوه بر این،Veo3 قابلیت تولید هم‌زمان صدا و گفتار هماهنگ با تصویر (Lip-Sync) را دارد. این ویژگی از طریق شبکه‌های کمکی مبتنی بر Audio Dif-fusion Models و الگوریتم‌های تطبیق حرکات لب با گفتار ممکن شده است. نتیجه، کلیپ‌هایی است که تشخیص مصنوعی بودن آن‌ها برای مخاطب دشوار است.

کارگردانی سینمایی با زبان طبیعی

یکی از برجسته‌ترین ویژگی‌های Veo3، توانایی آن در دریافت دستورات به زبان طبیعی برای کنترل عناصر

سینمایی است. برای مثال، با وارد کردن جمله‌ای مانند: «دوربینی با زاویه پایین، زنی را که در کوچه‌ای با نور نئون راه می‌رود دنبال کند»،
Veo3 نه‌تنها سوژه و محیط را بازسازی می‌کند، بلکه زاویه دوربین، عمق میدان، نورپردازی و حتی حس و حال صحنه را با دقتی خیره‌کننده پیاده‌سازی می‌کند. این قابلیت از طریق ماژول تکمیلی Flow ممکن شده است که با ترکیب فناوری‌های Veo،Imagen و Gemini، امکان طراحی صحنه، کارگردانی و تولید دارایی‌های بصری را در یک بستر یکپارچه فراهم می‌آورد.

کاربردهای پژوهشی و چالش‌های پیش‌رو

از منظر مهندسی،Veo3 دریچه‌ای نو به سوی پژوهش در حوزه‌های مدل های هم‌ترازی (Alignment Models)، انتشار زمانی (Temporal Diffusion) و بهینه‌سازی چندوجهی (Multimoda Fine-Tuning) گشوده است. هماهنگی دقیق صدا، تصویر و گفتار در این مدل، نیازمند الگوریتم‌های پیچیده هم‌آموزی است که می‌تواند الهام‌بخش نوآوری‌های آینده باشد. با این حال، چالش‌هایی مانند مصرف بالای انرژی محاسباتی، پتانسیل تولید محتوای جعلی (Deepfake) و مسائل مرتبط با شناسایی منشأ محتوا، پرسش‌های جدی در حوزه اخلاق و امنیت فناوری مطرح کرده‌اند. این موضوعات، پژوهشگران را به سوی تدوین چارچوب‌های نظارتی و توسعه ابزارهای تشخیص محتوای مصنوعی سوق داده است.

سخن پایانی

Veo3 بیش از یک ابزار تکنولوژیک، جهشی مفهومی در خلق محتواست. این مدل نه‌تنها برای مهندسان کامپیوتر فرصتی برای کاوش و توسعه فناوری‌های نوین فراهم می‌کند، بلکه با چالش‌های اخلاقی و فنی، زمینه‌ساز گفت‌وگوهای عمیق در جامعه علمی است. آینده تولید محتوا، جایی که داستان‌سرایی با چند خط متن به واقعیت تبدیل می‌شود، دیگر در افق نیست؛ اکنون در دسترس ماست، همراه با صدا، حرکت و داستان‌هایی که نفس می‌کشند.

حملات تزریق ؛ SQL تهدیدی خاموش در دل برنامه‌های وب

علی کاظم پور

برنامه‌های مبتنی بر وب، نوعی سیستم توزیع‌شده محسوب می‌شوند که در بسترهای متنوعی همچون مرورگرهای وب، سرورهای کاربردی و زیرساخت‌های شبکه‌ای اجرا می‌گردند. این برنامه‌ها طی سال‌های اخیر به‌طرز چشمگیری تحول یافته‌اند و به فناوری‌هایی کلیدی برای اجرای مؤثر فرآیندهای تجاری در محیط‌های سخت‌افزاری گوناگون تبدیل شده‌اند. رشد این حوزه به‌وضوح در آمارهای بازار بازتاب یافته است؛ به‌گونه‌ای که صنعت طراحی و توسعه وب، درآمدی چند میلیارد دلاری را به خود اختصاص داده است. همچنین، بهبود قابل توجه در تجربه کاربری موجب افزایش چشمگیر ترافیک وب شده و تعاملات کاربران عمدتاً از طریق دستگاه‌های همراه و در درجه بعد، رایانه‌های رومیزی صورت می‌گیرد.

با وجود تمامی مزایای برنامه‌های وب در فضای سازمانی امروز، این سامانه‌ها همچنان از آسیب‌پذیری‌های متعددی رنج می‌برند که در بسیاری از موارد توسط مهاجمان سایبری مورد سوءاستفاده قرار می‌گیرند. بر اساس گزارش‌های امنیتی، حدود ۴۳ درصد از نقض‌های امنیت سایبری ناشی از حملات مبتنی بر وب است. یکی از مهم‌ترین و خطرناک‌ترین تهدیدها در این زمینه، حملات تزریق SQL (SQL Injection Attack - SQLIA) است. در این نوع حمله، مهاجم با وارد کردن ورودی‌های مخرب در قالب پرس‌وجوهای SQL، تلاش می‌کند تا به‌صورت غیرمجاز به پایگاه داده دسترسی یابد. این دسترسی می‌تواند به افزایش سطح دسترسی و انجام اقدامات مخرب نظیر نشت اطلاعات حساس یا

نقض یکپارچگی داده‌ها منجر شود.

SQLIA در واقع یک آسیب‌پذیری نرم‌افزاری ناشی از نبود یا ضعف در سازوکارهای اعتبارسنجی ورودی است. از رایج‌ترین روش‌های مورد استفاده در این حملات، دست‌کاری مقادیر ورودی کاربر مانند نام کاربری و گذرواژه، در درون دستورات SQL است، به‌گونه‌ای که امکان دور زدن مکانیزم‌های احراز هویت و اجرای کوئری‌های مخرب فراهم گردد.

از آنجا که زبان SQL، ابزار اصلی مدیریت پایگاه داده‌ها در سمت سرور محسوب می‌شود، به‌شدت در معرض حملات تزریقی قرار دارد. مهاجمان با بهره‌گیری از آسیب‌پذیری‌های موجود در منطق برنامه، ساختار کوئری‌ها را به‌نحوی تغییر می‌دهند که بتوانند اطلاعات محرمانه را استخراج کرده و به داده‌های حفاظت‌شده دسترسی یابند. چنین تغییراتی، امکان اجرای دستورات ناخواسته را فراهم کرده و یکپارچگی و امنیت کل سامانه را به خطر می‌اندازد. افزایش چشمگیر حملات تزریق SQL، رابطه‌ای مستقیم با مهاجرت صنعت از پردازش‌های آفلاین به بسترهای آنلاین دارد. برنامه‌های تحت وب به‌واسطه حضور دائمی در فضای اینترنت، نسبت به سامانه‌های سنتی آفلاین هدف‌پذیری بیشتری دارند. این موضوع به‌ویژه در سامانه‌هایی که مبتنی بر پایگاه‌های داده رابطه‌ای بوده و با حجم بالایی از ترافیک مواجه هستند، سطح حمله را به‌شکل چشمگیری افزایش می‌دهد.

در بسیاری از سناریوهای رایج، مهاجمان با وارد کردن کاراکترها و دستورات غیرمجاز در کوئری‌های SQL، زمینه اجرای پرس‌وجوهای مخرب را فراهم می‌کنند. به‌عنوان نمونه، در فرآیند احراز هویت کاربر، در صورتی که ورودی‌هایی نظیر نام کاربری و رمز عبور به‌درستی اعتبارسنجی نشوند، ممکن است مهاجم بتواند با تزریق کدهای SQL، این فرآیند را دور بزند و به اطلاعات حساس، نظیر اعتبارنامه‌های مدیریتی، دست یابد.

انواع حملات تزریق SQL

حملات تزریق SQL در قالب‌های مختلفی بروز می‌یابند که از رایج‌ترین آن‌ها می‌توان به تزریق مبتنی بر خطا (Error-Based)، تزریق مبتنی بر تاتولوژی (Tautology-Based)، و تزریق مبتنی بر عبارت Union (Union-Based) اشاره کرد. این نوع حملات با ایجاد تغییرات غیرمجاز در ساختار پرس‌وجوهای SQL، امکان دست‌کاری، استخراج یا حذف اطلاعات پایگاه داده را برای مهاجم فراهم می‌سازند.

نکته قابل توجه آن است که آسیب‌پذیری در برابر حملات SQLIA تنها به پایگاه‌های داده رابطه‌ای مانند MySQL، PostgreSQL یا Oracle محدود نمی‌شود، بلکه پایگاه‌های داده غیررابطه‌ای نظیر MongoDB و Cassandra نیز با وجود تفاوت در مدل داده‌ای و زبان پرس‌وجو، در معرض چنین تهدیداتی قرار دارند.

حملات تاتولوژی (Tautology Attacks)

در این نوع از حملات، مهاجم با وارد کردن عباراتی که نتیجه‌ی آن‌ها همواره درست (true) ارزیابی می‌شود، تلاش می‌کند تا فرآیند احراز هویت را دور بزند و به پایگاه داده دسترسی غیرمجاز پیدا کند. این روش

معمولاً با افزودن یک شرط در بخش WHERE از کوئری SQL انجام می‌شود تا منطق پرس‌وجو را تغییر دهد.

در این حمله، استفاده از دستوری مانند(۱=۱) که همواره مقدار آن true است، باعث می‌شود شرط موجود در کوئری همواره برقرار باشد و در نتیجه محدودیت‌های احراز هویت بی‌اثر شوند.

در زیر، نمونه‌ای از یک کوئری SQL معتبر (با رنگ مشکی) در مقابل کوئری مخرب (با رنگ آبی) نشان داده شده است. کوئری مخرب با افزودن عبارت (WHERE ۱=۱) موفق می‌شود سیستم را فریب داده و احراز هویت را دور بزند.

```
SELECT * FROM user WHERE name
="xyz" AND password = "345cba"
```

```
SELECT * FROM user WHERE name
= xyz "AND password
""345cba"=OR "1" = "1
```

حملات مبتنی (Union-Based Attacks) (UNION)

در این نوع از حملات تزریق SQL، مهاجم یک کوئری نامعتبر را با کوئری اصلی ترکیب می‌کند و از عبارت UNION برای الحاق نتایج استفاده می‌نماید. هدف این حمله، افزودن نتایج دلخواه مهاجم به خروجی کوئری معتبر است، به‌گونه‌ای که منجر به دسترسی غیرمجاز به داده‌های اضافی شود.

به‌طور معمول، در این حمله مهاجم تلاش می‌کند ستون‌هایی مانند نام کاربری و گذرواژه را به نتیجه کوئری اصلی اضافه کند تا بتواند اطلاعات حساس کاربران را به‌دست آورد.

در زیر، نمونه‌ای از یک حمله مبتنی بر UNION نشان داده شده است که در آن، کوئری مخرب با افزودن ستون‌های نام کاربری و گذرواژه به خروجی کوئری معتبر، منجر به افشای اطلاعات محرمانه می‌شود.

```
SELECT m, n FROM table4 (authentic)
SELECT m, n FROM table4 UNION SELECT k.l
FROM table5 (union base query)
UNION  SELECT username, password FROM
username
```

حملات تزریق مبتنی بر خطا (Error-Based Injections)

در حالی که حملات مبتنی بر UNION از ترکیب چند کوئری SQL برای استخراج اطلاعات از طریق تغییر داده‌ها استفاده می‌کنند، حملات مبتنی بر خطا با بهره‌برداری از پیام‌های خطا تولیدشده توسط پایگاه داده، به‌طور غیرمستقیم مهاجم را به اطلاعاتی درباره ساختار داخلی پایگاه داده هدایت می‌کنند.

یکی از دلایل اصلی آسیب‌پذیری در برابر این نوع حمله، توسعه و تست برنامه‌های وب به‌صورت مستقیم بر روی سایت‌های زنده (Live) به‌جای محیط‌های آفلاین یا محدود است. در چنین شرایطی، خطاهای غیرعمدی می‌توانند اطلاعاتی حساس را در اختیار مهاجم قرار دهند و زمینه را برای دسترسی غیرمجاز و حملاتی مانند تزریق دستورات (Command Injections) فراهم کنند.

در مواردی که حمله‌ی تزریق مبتنی بر خطا رخ دهد، توسعه‌دهنده ممکن است با پیام‌های خطایی از سوی سیستم مواجه شود که شامل مواردی مانند خطای سرریز بافر (Buffer Overrun) ، مدیریت استثناها (Exception Catching) و اشتباهات قالب‌بندی رشته‌ها (Format String Errors) باشد؛ پیام‌هایی که می‌توانند برای مهاجم به‌منزله‌ی سرنخی از ساختار سیستم و پایگاه داده تلقی شوند.

حملات تزریق زنجیره‌ای (Piggy-Backed Injections)

در حملات تزریق زنجیره‌ای، مهاجم با استفاده از علائم جداکننده (Delimiters) مانند (;)، کدهای مخرب را به کوئری‌های اصلی متصل می‌کند و از این طریق، پرس‌وجوهای اضافی و مستقل را پس از کوئری اولیه اجرا می‌نماید. این پرس‌وجوهای افزوده‌شده می‌توانند برای دست‌کاری، حذف یا از بین بردن اطلاعات در پایگاه داده به کار روند و در نتیجه، موجب اختلال در عملکرد سیستم و کاهش قابلیت اطمینان آن شوند.

در زیر، نمونه‌ای از این نوع حمله نشان داده شده است که در آن، مهاجم با افزودن یک کوئری دوم پس از کاراکتر (;)، اقدام به حذف یک جدول از پایگاه داده می‌کند. این اقدام منجر به از دست رفتن اطلاعات و آسیب به ساختار پایگاه داده می‌شود.

```
SELECT * FROM user WHERE name
xyz" AND password ="
="345cba"; DROP table name
```

در مجموع، حملات تزریق SOL به‌عنوان یکی از آسیب‌پذیری‌های جدی در برنامه‌های وب، با بهره‌برداری از نقص‌های اعتبارسنجی ورودی، توانایی نفوذ به لایه‌های حساس پایگاه داده را دارند. با توجه به رشد روزافزون حجم داده‌ها و تنوع زیرساخت‌های نرم‌افزاری، مقابله با این نوع تهدید نیازمند طراحی دقیق سازوکارهای اعتبارسنجی، محدودسازی دسترسی‌ها و بازنگری مستمر در کدها است. تنها با درک عمیق از نحوه عملکرد این حملات و توجه ویژه به جزئیات پیاده‌سازی می‌توان از ورود غیرمجاز جلوگیری کرد. توسعه‌دهندگان باید فراتر از اصول اولیه امنیت حرکت کرده و با به‌کارگیری ابزارهای تخصصی و روش‌های نوآورانه، ساختارهای پایگاه داده و سامانه‌های وب را مقاوم‌تر کنند. در نهایت، امنیت یک هدف ثابت نیست بلکه فرآیندی است که با هر تغییر در تکنولوژی و تهدیدات، باید بازتعریف و به‌روزرسانی شود.

آیا بیل گیتس ایده CP/M را دزدید؟

نگاهی به یک جنجال تاریخی در دنیای کامپیوتر

محمد مهدی بابابیک

در تاریخ پر فراز و نشیب کامپیوترهای شخصی، داستان‌های بسیاری از نوآوری، رقابت و البته جنجال وجود دارد. یکی از بحث‌برانگیزترین این داستان‌ها، ماجرای ظهور سیستم‌عامل MS-DOS از مایکروسافت و اتهاماتی است که همواره علیه بیل گیتس مبنی بر «دزدیدن» ایده اصلی سیستم‌عامل CP/M مطرح بوده است. این مقاله به بررسی شواهد، روایت‌های مختلف و تحلیل فنی این موضوع می‌پردازد تا به درک عمیق‌تری از این واقعه تاریخی دست یابیم.

بازیگران اصلی صحنه

برای فهم دقیق این ماجرا، ابتدا باید با شخصیت‌های کلیدی آن آشنا شویم:

- گری کیلْدال (Gary Kildall):موسس شرکت Digital Research Inc. (DRI) و خالق سیستم‌عامل CP/M (Control Program for Microcomputers).کیلْدال یک دانشمند کامپیوتر برجسته و از پیشگامان صنعت نرم‌افزار بود.
- بیل گیتس (Bill Gates): موسس جوان و جاه‌طلب مایکروسافت که در آن زمان به خاطر زبان برنامه‌نویسی بِیسیک (BASIC) شهرت داشت.
- تیم پترسون (Tim Paterson): برنامه‌نویسی در شرکت Seattle Computer Productsکه سیستم‌عاملی به نام QDOS(Quick and Dirty Operating System) را توسعه داد.

IBM: غول دنیای کامپیوتر که در اواخر دهه ۱۹۷۰ تصمیم گرفت وارد بازار کامپیوترهای شخصی (PC) شود و به یک سیستم‌عامل ۱۶ بیتی نیاز فوری داشت.

روایت ماجرا: از یک جلسه سرنوشت‌ساز تا یک قرارداد تاریخی

در سال ۱۹۸۰، نمایندگان IBM برای تأمین سیستم‌عامل کامپیوتر شخصی جدید خود به سراغ مایکروسافت رفتند. در آن زمان، مایکروسافت سیستم‌عامل نداشت اما بیل گیتس که فرصت را دریافته بود، آن‌ها را به سمت شرکت Digital Research و سیستم‌عامل CP/M که استاندارد غیررسمی سیستم‌های ۸ بیتی بود، راهنمایی کرد.

روایت‌ها در مورد ملاقات نمایندگان IBM و گری کیلْدال متفاوت است. مشهورترین روایت که توسط منابع نزدیک به مایکروسافت ترویج شده، حاکی از آن است که کیلْدال در روز ملاقات با همسرش دوروتی مک‌ایون (که شریک تجاری‌اش نیز بود) برای یک پرواز تفریحی با هواپیمای شخصی‌اش رفته بود و مذاکرات اولیه با وکیل شرکت به نتیجه نرسید. روایت دیگر که از سوی DRI مطرح شده، می‌گوید کیلْدال در حال انجام کارهای تجاری دیگری بوده و مذاکرات به دلیل عدم توافق بر سر قرارداد عدم افشا (NDA) با IBM به بن‌بست خورده است.

هرچه که بود، IBM دست خالی به سراغ بیل گیتس بازگشت. گیتس که نمی‌خواست این فرصت طلایی را از دست بدهد، قول تأمین یک سیستم‌عامل را به IBM داد، در حالی که هنوز محصولی در دست نداشت.

ورود QDOS و تولد MS-DOS

بیل گیتس به سرعت به دنبال یک سیستم‌عامل ۱۶ بیتی گشت و گزینه‌ی خود را در سیاتل یافت: ODOS که بعدها به ۸۶-DOS نیز معروف شد. این سیستم‌عامل توسط تیم پترسون در شرکت SCP برای پردازنده‌های جدید ۱۶ بیتی اینتل (۸۰۸۶) نوشته شده بود. پترسون برای نوشتن ODOS از مستندات CP/M الهام گرفته بود تا برنامه‌های نوشته شده برای CP/M به راحتی به سیستم‌عامل جدید منتقل شوند.

مایکروسافت ابتدا با پرداخت ۲۵,۰۰۰ دلار، حق بازاریابی ODOS را از SCP خرید و سپس در معامله‌ای نهایی به ارزش ۵۰,۰۰۰ دلار، مالکیت کامل آن را به دست آورد. بیل گیتس و تیمش، این سیستم‌عامل را برای کامپیوتر شخصی IBM بهینه‌سازی کرده و نام آن را به PC-DOS (برای IBM) و MS-DOS (برای فروش به سایر شرکت‌ها) تغییر دادند.

قرارداد مایکروسافت با IBM یک شاهکار تجاری بود. مایکروسافت به ازای هر کامپیوتر فروخته شده توسط IBM، مبلغی به عنوان حق امتیاز دریافت می‌کرد، اما مهمتر از آن، حق فروش MS-DOS به سایر سازندگان کامپیوترهای سازگار با IBM را برای خود حفظ کرد. این استراتژی، مایکروسافت را به بازیگر اصلی بازار سیستم‌عامل‌ها و بیل گیتس را در مسیر تبدیل شدن به یکی از ثروتمندترین افراد جهان قرار داد.

آیا این یک سرقت بود؟ تحلیل فنی و حقوقی

ادعای اصلی گری کیلْدال و شرکت DRI این بود که MS-DOS یک کپی مستقیم از CP/M است. اما آیا این ادعا از نظر فنی و حقوقی قابل اثبات است؟

شباهت‌های فنی:

- رابط برنامه‌نویسی کاربردی (API): تیم پترسون آگاهانه API سیستم‌عامل خود را بسیار شبیه به CP/M طراحی کرد. فراخوانی‌های سیستمی (System Calls) در ODOS (که از طریق وقفه نرم‌افزاری INT 21h انجام می‌شد) شباهت زیادی به فراخوانی‌های BDOS (Basic Disk Operating System) در CP/M (که از طریق CALL 5 صورت می‌گرفت) داشت. این شباهت، فرآیند پورت کردن (انتقال) نرم‌افزارها از CP/M به MS-DOS را بسیار آسان می‌کرد و یک مزیت رقابتی بزرگ برای مایکروسافت بود.
- ساختار فرمان و نام‌گذاری فایل‌ها: ساختار خط فرمان و قواعد نام‌گذاری فایل‌ها (۸ کاراکتر برای نام و ۳ کاراکتر برای پسوند) نیز مستقیماً از CP/M الگوبرداری شده بود.

تفاوت‌های فنی:

- سیستم فایل (File System): این یکی از بزرگترین تفاوت‌ها بود. MS-DOS از سیستم فایل FAT (File Allocation Table) استفاده می‌کرد که توسط خود بیل گیتس و مارک مک‌دونالد در سال‌های قبل برای زبان بیسیک طراحی شده بود. این سیستم فایل نسبت به ساختار ساده‌تر CP/M کارآمدتر بود.
- ساختار داخلی: با وجود شباهت در API، کد منبع (Source Code) دو سیستم‌عامل متفاوت بود. پترسون همواره اصرار داشت که ODOS را از ابتدا و بدون دسترسی به کد CP/M نوشته است.

جنبه حقوقی و اخلاقی

در دهه ۱۹۸۰، قوانین مربوط به مالکیت معنوی نرم‌افزار به وضوح امروز نبود. مفهوم «مهندسی معکوس پاک» (Clean-room reverse engineering) که در آن یک تیم بدون دسترسی به کد منبع، عملکرد یک نرم‌افزار را بازسازی می‌کند، یک رویه حقوقی پذیرفته شده بود. تیم پترسون ادعا کرد که تنها از روی مستندات و راهنماهای کاربری CP/M که به صورت عمومی در دسترس بودند، برای ساخت یک سیستم‌عامل سازگار استفاده کرده است.

با این حال، از نظر

اخلاقی، بسیاری معتقدند که مایکروسافت از ایده‌ها و تلاش‌های چندین ساله گری کیلْدال و DRI بهره‌برداری تجاری کرده است. کیلْدال خود را قربانی یک مانور تجاری هوشمندانه اما غیرمنصفانه می‌دید.

نتیجه‌گیری: دزدی یا هوش تجاری؟

پاسخ به این سوال که «آیا بیل گیتس ایده CP/M را دزدید؟» ساده نیست و بستگی به زاویه دید دارد.

- از دیدگاه حقوقی: با توجه به قوانین آن زمان و تفاوت‌های موجود در کد منبع، اثبات سرقت مستقیم (کپی‌برداری خط به خط) تقریباً غیرممکن است. مایکروسافت هرگز به خاطر نقض کپی‌رایت CP/M محکوم نشد.
- از دیدگاه فنی: MS-DOS به وضوح یک «کلون» یا «همسان» از CP/M در سطح عملکرد و API بود، اما در لایه‌های زیرین تفاوت‌های مهمی داشت. نمی‌توان آن را یک کپی صرف دانست، بلکه یک پیاده‌سازی مجدد با الهام بسیار زیاد است.
- از دیدگاه تجاری و اخلاقی: بیل گیتس فرصتی را شناسایی کرد که گری کیلْدال آن را از دست داد یا دست کم گرفت. او با هوش تجاری بالا، محصولی موجود (ODOS) را خرید، آن را به سرعت برای یک مشتری بزرگ (IBM) آماده کرد و با یک قرارداد استثنایی، آینده شرکتش را تضمین نمود. در این فرآیند، او قطعاً از کار و ایده اولیه رقیب خود بهره برد.

شاید بهترین توصیف برای این واقعه، ترکیبی از فرصت‌طلبی هوشمندانه بیل گیتس، عدم قاطعیت و شاید بدشانسی گری کیلْدال، و ابهامات حقوقی دنیای نوپای نرم‌افزار باشد. MS-DOS ممکن است یک «دزدی» به معنای رایج کلمه نباشد، اما داستان آن نمونه‌ای کلاسیک از رقابت بی‌رحمانه در دره سیلیکون است که در آن، سرعت، هوش تجاری و گاهی بی‌پروایی، به اندازه نوآوری فنی اهمیت دارد. این ماجرا، بیش از آنکه داستان یک سرقت باشد، داستان چگونگی تغییر مسیر تاریخ تکنولوژی توسط یک تصمیم تجاری است.

همکاری در نشریه ی ماتریس

نشریه ی ماتریس در جهت ارتقای کیفیت نشریه و مشارکت همه دانشجویان در سه تیم تحریریه، ویراستاری و طراحی گرافیک عضو همکار می پذیرد.

جهت ارتباط با روابط عمومی نشریه و همکاری در تهیه نشریه با ما در ارتباط باشید.

@MatrisMagazine