รายงาน

เรื่อง การออกแบบ Data Pipeline สำหรับตลาด Cryptocurrency

เสนอ

ผศ. ดร.อารีรัตน์ ตรงรัศมีทอง และ ผศ. ดร.เสมอแข สมหอม

จัดทำโดย นักศึกษาปริญญาโท ชั้นปีที่1

1. นาย อภิชาติ กันสีนวล
2. นาย นนทกุล เพชรรู้แจ้ง
3. นาย ณัฐภัทร ศิริพินทุ์

รายงานเล่มนี้เป็นส่วนหนึ่งของกระบวนวิชา 204721 – Data Engineering

ภาคเรียนที่ 1 ปีการศึกษา 2565

คณะวิทยาศาสตร์ ภาควิชาวิทยาการคอมพิวเตอร์

มหาวิทยาลัยเชียงใหม่

คำนำ

รายการเล่มนี้ เกิดขึ้นเพื่อประเมินความรู้ความเข้าใจที่ได้รับจากกระบวนวิชา 204721 – Data Engineering โดยทางทีมได้รับมอบหมายจาก ผศ. ดร.อารีรัตน์ ตรงรัศมีทอง และ ผศ. ดร.เสมอแข สมหอม ให้ออกแบบ Data pipeline ที่สามารถนำมาใช้งานได้จริง และสอดคล้องกับภาคทฤษฎีที่ได้ร่ำเรียนมา

ซึ่งทางทีมนั้น ได้มีความสนใจในตลาด Cryptocurrency เป็นพิเศษ และต้องการสร้างเครื่องมือที่ช่วยพยากรณ์สัญญาณซื้อ-ขาย ที่แม่นยำ และน่าเชื่อถือ ให้แก่นักลงทุนในตลาด Cryptocurrency ดังนั้นทางทีมจึงได้เริ่มสำรวจปัญหาต่างๆ ที่เกิดขึ้นในตลาดนี้ และพบว่าปัญหาส่วนใหญ่ที่ทำให้ สัญญาณซื้อ-ขาย ไม่ค่อยแม่นยำ หรือไม่ได้รับความเชื่อถือนั้นเกิดจากแหล่งข้อมูลต้นทางที่ไม่ครอบคลุมปริมาณซื้อ-ขายจริงของตลาดโลก, ต้นทุนการเข้าถึงเครื่องมือต่างๆที่ค่อนข้างสูง, รูปแบบการวิเคราะห์แนวโน้มตลาดที่ไม่ชัดเจน และแหล่งจัดเก็บข้อมูลปลายทาง รวมถึงการแสดงผลที่ยังห่างไกลความน่าเชื่อถือ

ดังนั้น ทางทีมจึงมุ่งมั่นศึกษาค้นคว้าองค์ประกอบต่างๆ ที่จำเป็นต่อการสร้าง Data pipeline สำหรับตลาด Cryptocurrency โดยเลือก Forecasting model เพื่อตอบรับกับเป้าหมายการสร้างเครื่องมือที่ช่วยพยากรณ์สัญญาณซื้อ-ขาย ซึ่งทางทีมหวังว่ารายงานเล่มนี้จะเป็นประโยชน์ทั้งในแง่การแลกเปลี่ยนความรู้ และเป็นเครื่องมือที่ช่วยเพิ่มศักยภาพในการลงทุนให้แก่นักลงทุนด้วย

คณะผู้จัดทำโครงการ
นาย อภิชาติ กันสีนวล
นาย นนทกุล เพชรรู้แจ้ง
นาย ณัฐภัทร ศิริพินทุ์

สารบัญ

วัตถุประสงค์ของโครงการ

ในโลกการเงินปัจจุบันนั้นเผชิญความท้าทายมาก เนื่องจากในช่วงสิบกว่าปีที่ผ่านมา มีการคิดค้นนวัตกรรมทางด้านการเงินที่หลากหลาย ซึ่งเข้ามาเปลี่ยนแปลงภาพลักษณ์ของโลกแห่งการเงิน-การลงทุนอย่างสิ้นเชิง โดยเฉพาะอย่างยิ่งการกำเนิดของสิ่งที่เรียกว่า Cryptocurrency หรือเงินดิจิตอลนั้น ได้เปลี่ยนแปลงโลกการเงินและเทคโนโลยีไปอย่างสิ้นเชิง ทั้งในแง่ของแนวคิดและวิธีการลงทุน ที่ต้องการความรวดเร็ว การติดตามการเปลี่ยนแปลงของราคาซื้อ-ขายอย่างต่อเนื่อง 24 ชั่วโมง รวมถึงปริมาณข้อมูลของเหรียญ Cryptocurrency ที่มีจำนวนที่มาก ทำให้เกิดความยากแก่นักลงทุนในการติดตามการเปลี่ยนแปลงของราคา, การค้นหาให้จังหวะในซื้อ-ขาย รวมถึงการส่งคำสั่งซื้อ-ขาย

ดังนั้นในโครงการค้นคว้านี้จึงมีจุดประสงค์ในการสร้าง Data Pipeline ของ Automated Trading Bot ที่จะเป็นส่วนช่วยให้การลงทุนของนักลงทุนใน Cryptocurrency มีความสะดวกรวดเร็วและมีประสิทธิภาพมากยิ่งขึ้น แถมยังช่วยลดภาระในการส่งคำสั่งซื้อ-ขายอีกด้วย โดยในโครงการค้นคว้าครั้งนี้ จะตอบคำถามว่าในการสร้าง Data Pipeline ของ Automated Trading Bot นั้น จะต้องเผชิญความท้าทายและต้องแก้ปัญหาเหล่านั้นด้วยเทคโนโลยีอะไร และมีโครงสร้างของ Data Pipeline อย่างไร โดยคำถามที่ Data Engineer จะต้องตอบคำถามมีดังนี้

1. จะใช้ภาษาอะไรในการทำ Data Pipeline?
2. ข้อมูลราคาของ Cryptocurrencyจะนำมาจากแหล่งไหน?
3. จะใช้ตัวช่วยอะไรในการจัดการและประมวลผลข้อมูล?
4. การจัดเก็บข้อมูลอย่างไร?
5. กลยุทธของการซื้อ-ขายคืออะไร?
6. จะส่งคำสั่งซื้อ-ขายอย่างไร?
7. เครื่องมือที่ช่วยในการจัดการ Data Pipeline คืออะไร?
8. ตัวช่วยประมวลผลการดำเนินการควรเป็นอะไร ระหว่าง On-Cloud vs On-Premise?

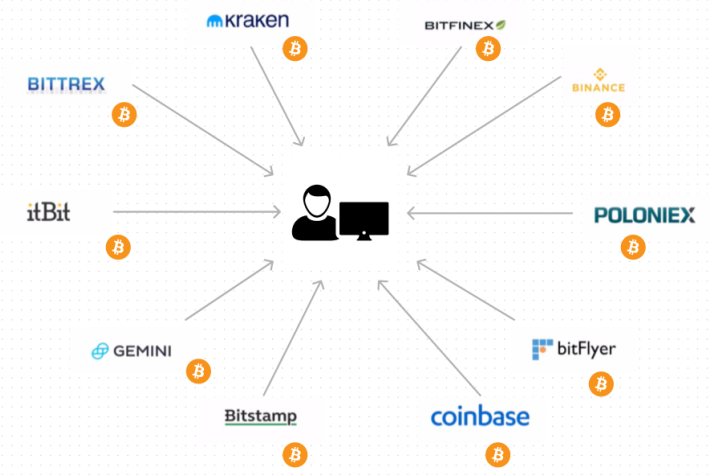
ภาษาที่ใช้ในโครงการ

งานวิจัยนี้ทางทีมได้ศึกษาภาษาต่างๆ เพื่อมาเปรียบเทียบถึงโอกาส ความเป็นไปได้และข้อดีข้อเสียของแต่ละภาษา เพื่อเลือกภาษาที่เหมาะสมกับจุดประสงค์ของโครงการค้นคว้านี้ โดยพบว่า Python เป็นภาษาที่เหมาะมากที่สุดในการสร้าง Data Pipeline สำหรับระบบ Automated Trading Bot ให้แก่นักลงทุนในการลงทุน Cryptocurrency

โดย Python นับเป็นภาษาโปรแกรมที่ได้รับยอดนิยมในปัจจุบัน และมักถูกนำไปใช้ในหลายประเภทงาน อาทิ สร้างโมเดลของ Machine Learning, สร้าง Web Application และยังนิยมนำมาใช้งานเพื่อวัตถุประสงค์ Automation ได้ด้วย อาทิ การสร้าง Automating Tasks และ Data Pipeline เป็นต้น Python นั้นเป็น High-level Programming Language ที่ใช้งานง่าย สามารถเขียน script ที่ยืดหยุ่นเพื่อรองรับการขยายตัวของข้อมูลและการเปลี่ยนแปลงของโมเดลธุรกิจในอนาคตได้ รวมถึงความง่ายในการศึกษาเบื้องต้น ทำให้ง่ายต่อการดูแลรักษาและส่งต่อให้ทีมพัฒนาในอนาคตอีกด้วย

ในแง่การทำ Data Pipeline นั้น Python นิยมถูกใช้ในการทำ Big Data Processing เนื่องจากมันสามารถดาวน์โหลดข้อมูล, นำส่งข้อมูล และทำความสะอาดข้อมูลได้อย่างมีประสิทธิภาพ จึงเป็นภาษาที่บริษัทฯส่วนใหญ่ที่ดำเนินการเกี่ยวกับ Big Data ได้ให้ความไว้วางใจในการนำมาใช้งาน อีกทั้ง Python นั้นยังมี Library ที่หลากหลายและเป็นตัวช่วยให้ผู้พัฒนาสามารถนำมาเป็นตัวช่วยในใช้งานหลากหลาย ตั้งแต่ การคำนวณ, การวิเคราะห์ข้อมูล รวมถึงการทำ Data Processing นอกจากนี้ Python เป็นภาษาที่สามารถทำงานร่วมกับเทคโนโลยียอดนิยมที่นำมาใช้ในการประมวลผลเกี่ยวกับ Big Data เช่น Apache Spark และ Apache Air Flow ได้อีกด้วย

แหล่งข้อมูลที่นำมาใช้ในโครงการ



คุณภาพของข้อมูล Cryptocurrencyเป็นหนึ่งความท้าทายของข้องมูลในการทำ Automated Trading Bot เนื่องจากคุณลักษณะของ Cryptocurrencyนั้น เป็นสิ่งที่มีการเก็บข้อมูลแบบกระจายศูนย์ทั้งในด้านของการถือครอง รวมถึงมี Exchange ซึ่งเป็นสถานที่ในการให้บริการซื้อขาย Cryptocurrency ก็มีให้บริการหลายที่เช่นกัน ด้วยเหตุนี้ทำให้ราคาซื้อ-ขายของแต่ละตลาดนั้นไม่เท่ากัน ยกตัวอย่างเช่นราคาซื้อ-ขาย Bitcoin ใน Binance อาจจะอยู่ที่ 20,000 USD ต่อ 1 BTC ในขณะที่ใน itBit ซึ่งมีปริมาณสภาพคล่องน้อยกว่า ทำให้มีความเป็นไปได้ว่าราคาซื้อ-ขายมีโอกาสที่จะสูงกว่า หรือต่ำกว่า 20,000 USD ต่อ 1 BTC ก็เป็นไปได้

ดังนั้นการหาราคากลางในการซื้อ หรือการขาย Cryptocurrency นับเป็นหนึ่งในความท้าทายหลักในการสร้าง Automated Trading Bot ที่น่าเชื่อถือ ที่สามารถนำมาเป็นตัวแทนของข้อมูล Cryptocurrencyได้ ยิ่งไปกว่านั้นข้อมูลคู่เงินที่ใช้แลกเปลี่ยนก็เป็นอีกหนึ่งความท้าทายเพราะ Cryptocurrency สามารถแลกเปลี่ยนได้ด้วยเงิน Fiat Currency เช่น BTC/USD, BTC/JPY, BTC/THB, ฯลฯ รวมถึงเป็น Cryptocurrency ด้วยกันก็ได้เช่นกัน อาทิ BTC/ETH, BTC/LUNA เป็นต้น

นอกจากนี้การเข้าถึงข้อมูลที่เกี่ยวกับ Cryptocurrency ของแต่ละ Exchange นั้นก็เป็นอีกหนึ่งความท้าทาย เนื่องจากแต่ละ Exchange ก็มีรูปแบบของข้อมูลที่แตกต่างกัน ซึ่งบาง Exchange ก็มีบริการให้ข้อมูลผ่าน API ในขณะที่บาง Exchange ก็ไม่มีการให้บริการด้านการให้ข้อมูลซึ่งนี่เป็นอีกหนึ่งข้อจำกัดในการรวบรวมราคาที่เพื่อนำมาตัวแทน

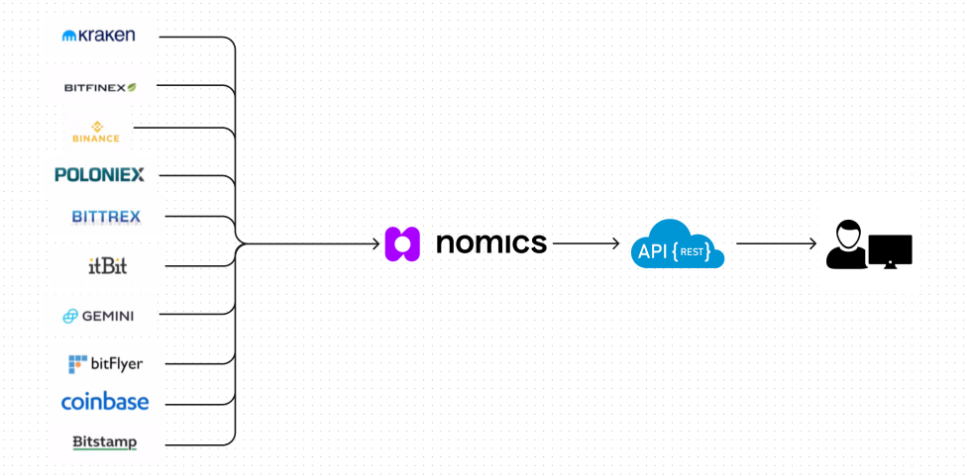
ดังที่กล่าวมาข้างต้นทางทีมจึงได้หาผู้ให้บริการข้อมูล Cryptocurrency เพื่อนำมาพิจารณาช่องทางในการเข้าถึงข้อมูล Cryptocurrencyเพื่อนำมาเป็นข้อมูลตั้งต้นในการทำ Automated Trading Bot โดยด้านล่างคือผู้ให้บริการข้อมูล Cryptocurrency ที่ให้บริการและมีความเป็นไปได้ในการเป็น Source of Data สำหรับทำ Automated Trading Bot ของโครงการค้นคว้านี้

1. Nomics API

Nomics เป็นผู้ให้บริการข้อมูลตลาด cryptocurrency และ Bitcoin จากตลาด Exchange ต่างๆให้แก่นักลงทุนและผู้ใช้งาน โดย Nomics นั้นก่อตั้งขึ้นเมื่อปี ค.ศ. 2018 จึงมีข้อมูลราคาและปริมาณการซื้อ-ขาย Cryptocurrency เหรียญต่างๆรวมถึงเครื่องมือหลากหลายให้ผู้ใช้งานเลือกใช้ได้ในด้านต่างๆ อาทิ การสร้างพอร์ตการลงทุน, การทำApplication, การทำกลยุทธการซื้อ-ขาย Cryptocurrency รวมถึงการสร้าง Robot เพื่อทำการซื้อ-ขาย Cryptocurrency เป็นต้น ในด้านของข้อมูล Nomics นั้นเก็บข้อมูลการซื้อ-ขายของเหรียญ Cryptocurrencyต่างๆ จาก Exchange มากกว่า 822 แห่งทั่วโลก ซึ่งครอบคลุมปริมาณการซื้อของมากกว่า 80% ของโลก และทำการรวมข้อมูล (Data Aggregation) และประมวลผลข้อมูล โดยผลลัพธ์ก็คือข้อมูลซึ่งทำหน้าที่เป็นตัวแทนของ Cryptocurrency เหรียญนั้นๆ ที่มีความน่าเชื่อถือและใกล้เคียงราคาซื้อ-ขายที่เป็นจริงมากที่สุด

ปัจจุบันนั้นข้อมูลของปริมาณการซื้อขาย Cryptocurrency บน Nomics นั้นมีทั้งหมด 78 ล้านเทรด ซึ่งมีปริมาณการเรียกใช้ API ของ Nomics จากผู้ใช้งานประมาณ 321 ล้านครั้งต่ออาทิตย์ และมีเวลาเฉลี่ยในการตอบรับของ API นั้นอยู่ที่ 0.51 วินาทีเท่านั้น นอกจากนี้ Nomics API ยังอนุญาตให้บริการพิเศษต่างๆสำหรับผู้ใช้งานที่จ่ายเงิน โดยสามารถการปรับรูปแบบของข้อมูลให้สามารถตอบสนองการใช้งานของแต่ละผู้ใช้งานได้ รวมถึงบริการส่งข้อมูลที่มีความละเอียดสูงให้กับผู้ใช้งานอีกด้วย โดย Nomics API นั้นไม่มีข้อจำกัดในการเรียกใช้งานดังนั้นจึงการันตีได้ว่าผู้ใช้งานและนักพัฒนาสามารถใช้งานได้ต่อเนื่อง

ในด้านการให้บริการ Nomics มุ่งเน้นการให้บริการผ่าน Restful API เป็นหลัก โดยมีเอกสารและตัวอย่างประกอบการใช้งานเผยแพร่บนเว็บไซต์ให้แก่ผู้ใช้งานและนักพัฒนาสามารถดูได้ รวมถึงมีตัวอย่าง Code ในการเชื่อมต่อ Nomics API กับโปรแกรมในภาษาต่างๆ อาทิ Python JavaScript และ Ruby เป็นต้น สุดท้ายนี้ Nomics API มีทีม Support ผู้ใช้งาน 24 ชั่วโมงในกรณีที่มีปัญหา ทางทีมงานนักพัฒนาของ Nomics สามารถให้คำแนะนำแก่ผู้ใช้งานเพื่อแก้ปัญหาให้ได้อย่างทันท่วงทีอีกด้วย



ข้อดี

- ใช้งานฟรี
- มุ่งเน้นการให้บริการข้อมูลผ่าน API เป็นหลัก
- มีประวัติข้อมูลของแทบทุก Cryptocurrencyบนโลก
- ข้อมูลมีความน่าเชื่อถือและใกล้เคียงกับราคาซื้อ-ขายจริงของ Cryptocurrency
- Support การแลกเปลี่ยนหลากหลายสกุลเงิน
- มีข้อมูลแท่งเทียน Candlestick ได้ด้วยการใช้ API
- เครื่องมือมากมาย ช่วยในการประเมินพอร์ตการลงทุน
- การันตีการทำงานต่อเนื่อง และไม่จำกัดปริมาณการเรียกใช้งาน API
- ตอบสนองไว

ข้อเสีย

- เสียค่าใช้จ่ายในการสมัครบริการเพื่อรับข้อมูล
- เน้นการให้บริการผ่าน API เป็นหลัก ทำให้การเรียกใช้งานข้อมูลผ่านรูปแบบ Batch ไฟล์อาจจะไม่สามารถทำได้

2**. CryptoCompare API**

CryptoCompare ผู้บริการข่าวสาร และข้อมูลของ Cryptocurrency โดยผู้ใช้งานสามารถดึงข้อมูลผ่าน API ของ CryptoCompare ได้ โดยข้อมูลของ Cryptocurrency ของ CryptoCompare นั้นครอบคลุมตลาดซื้อ-ขาย 311 แห่งทั่งโลก และครอบคลุม Cryptocurrency 7,490 เหรียญ ซึ่งมีการเรียกใช้งาน 40,000 ครั้งต่อวินาที

ข้อดี

- มีการให้บริการการดึงข้อมูลผ่านระบบ REST API และ Batch File
- ข้อมูล historical data รายชั่วโมง มีให้เข้าย้อนดูได้เป็นปี
- ราคาค่าบริการ ค่อนข้างดีกว่าเจ้าอื่น
- มีข้อมูลข่าวรายวันของ CryptoCompare

ข้อเสีย

- ในกรณีที่ต้องการข้อมูลเฉพาะจะต้องจ่ายเงินเพิ่ม
- คุณภาพการให้บริการไม่คงที่ และไม่มีระบบ Customer Support

ตารางเปรียบเทียบผู้ให้บริการข้อมูล Cryptocurrency

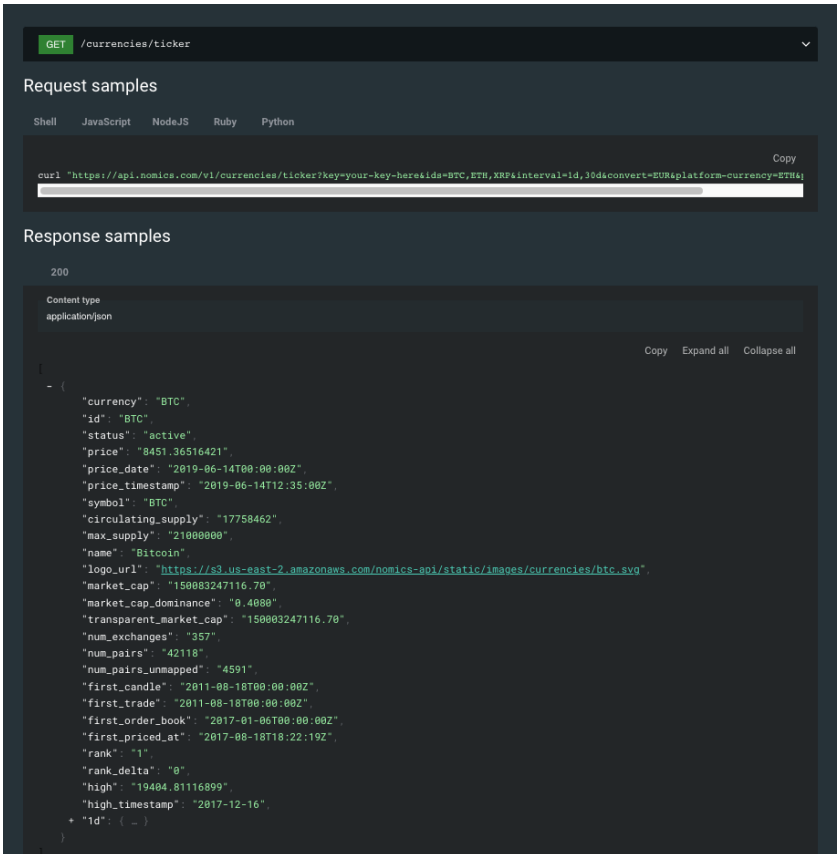
	Nomics	CryptoCompare
REST API	✓	✓
WebSocket API	✓	✓
FIX API	✗	✗
Rates bottleneck (requests/second)	Unlimited	1.67
Can fetch data with formulas	✓	✓
Ticker level data	✓	✓
Best bid/ask (Level 1 Data)	✓	✓
Trade level data	✓	✗
Historical trade level data	✓	✓
Order book data (Level 2 Data)	✓	✗
Historical order book data	✗	✗
All time high data	✓	✗
Currency conversion	✓	✓
Historical currency exchange rate	✓	✗
OHLC historical data	✓	✓
OHLC smallest interval	1 minute	1 minute
SLAs	✓	✗
Get support by	Phone/ / Telegram / priority e-mail	e-mail
Support forum	✓	✓
Test environment	✗	✗
Integration library	✗	✗
Integration examples	✗	✗
Quality of the documentation	Good	Below average
Markets	885	885

จากที่กล่าวมาข้างต้นเนื่องจาก Nomics เก็บข้อมูลการซื้อขายของเหรียญ Cryptocurrency ต่างๆจาก Exchange ได้มากกว่า 822 แห่งทั่วโลก ซึ่งครอบคลุมปริมาณการซื้อ-ขายมากกว่า 80% ของโลก จึงทำให้ข้อมูลเหล่านี้สามารถเป็นตัวแทนของ Cryptocurrency เหรียญนั้นๆ ได้อย่างมีความน่าเชื่อถือและใกล้เคียงราคาซื้อ-ขาย ที่เป็นจริงมากที่สุด นอกจากนี้ Nomics มุ่งเน้นการให้บริการผ่าน Restful API เป็นหลักโดยมีเอกสารและตัวอย่างประกอบการใช้งานเผยแพร่บนเว็บไซต์ให้แก่ผู้ใช้งานและนักพัฒนาสามารถดูได้ รวมถึงมีตัวอย่าง Code ในการเชื่อมต่อ Nomics API กับโปรแกรมในภาษาต่างๆ และระบบ Support ผู้ใช้งาน ตลอด 24 ชั่วโมง ดังนั้นด้วยเหตุผลทั้งหลายนี้ จึงทำให้ทางทีมเลือกใช้ Nomics เป็น Source of data สำหรับการทำ Automated Trading Bot ของโครงการนี้

ตัวอย่าง การขอข้อมูลที่ใช้ในโครงการค้นคว้าครั้งนี้

Data Pipeline ที่จะเป็นตัวช่วยนักลงทุนในการหาจังหวะในการซื้อ-ขาย ได้แก่ข้อมูล Currencies Ticker ของ Nomics API ซึ่งเป็นข้อมูลที่เกี่ยวข้องกับเหรียญ Cryptocurrency แต่ละเหรียญ โดยในรายละเอียดข้อมูลของเหรียญนั้นๆ จะประกอบไปด้วยข้อมูลต่างๆที่เกี่ยวข้อง อาทิ ราคา, Symbol, มูลค่าตลาด, ประมาณการซื้อ-ขาย, ข้อมูล ณ ช่วงเวลาใด และอื่นๆ ดังที่เห็นในภาพด้านล่างเป็นต้น

โดยในการค้นคว้าครั้งนี้ ทางทีมเลือกใช้ข้อมูล Ticker ซึ่งเป็นข้อมูลราคาของ Cryptocurrency โดยสาเหตุที่เลือกใช้ Ticker ก็เพราะเป็นข้อมูล Snap shot ของราคา Cryptocurrency ณ เวลาที่ส่ง Request โดย Time Frame ในการดึงข้อมูลก็จะสอดคล้องกับความถี่ของกลยุทธ์การซื้อ-ขาย โดยถ้าต้องการทำ Daily Trading ก็ดึงข้อมูลวันละครั้ง แต่ถ้าต้องการทำ Intraday Trading ก็สามารถดึงข้อมูลได้ตามความถี่ที่ต้องการซึ่งจะสอดคล้องกับกลยุทธ์การลงทุนของแต่ละผู้ใช้งาน เช่นระดับชั่วโมงหรือระดับนาที เป็นต้น



จากภาพด้านบนข้อมูล Cryptocurrency ที่สนใจจะถูกส่งข้อมูลกลับมาใน Format ข้อมูล JSON ผ่าน Restful API โดย Response ของ Ticker Request โดยจะมีขนาดประมาณ 226 KB ต่อหนึ่งไฟล์โดยเราสามารถคาดการณ์ได้ว่าถ้าทำ Daily Trading จะมีขนาดของข้อมูลประมาณ 82,480 KB หรือ 82 MB และถ้าทำ Intraday Trading (Hourly) จะมีขนาดของข้อมูลประมาณ 1 GB และถ้าทำ Intraday Trading (5 Mins) จะมีขนาดของข้อมูลประมาณ 11.5 GB ทั้งนี้ความถี่ของการดึงข้อมูลก็จะสอดคล้องกับกลยุทธ์การเทรดของนักลงทุนแต่ละคนกันไป

ตัวช่วยในการจัดการและประมวลผลข้อมูล

การประมวลผลข้อมูลนั้น ถือเป็นสิ่งที่ต้องคำนึงถึงหลังจากได้รับข้อมูลที่ต้องการใช้มาแล้ว โดยในโครงการค้นคว้าครั้งนี้ทางทีมได้เลือกสองเทคโนโลยีมาช่วยในการจัดการข้อมูลและประมวลผลข้อมูลได้แก่

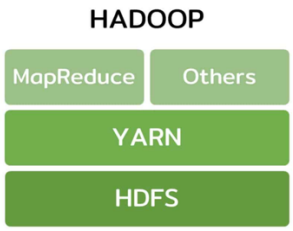
1. HADOOP
2. Apache Spark

โดยรายละเอียดของแต่ละเทคโนโลยีมีดังนี้

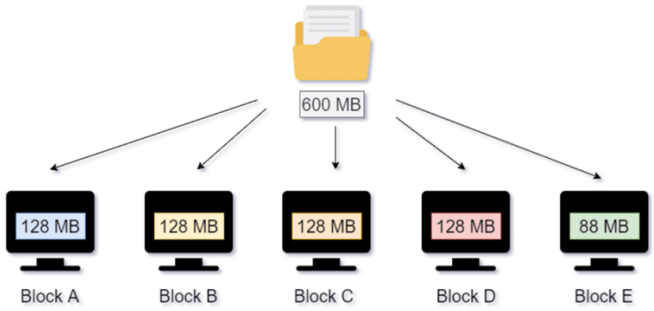
1. HADOOP

HADOOP เป็นระบบจัดเก็บข้อมูลที่สามารถจัดการกับความหลากหลายของข้อมูลได้ เช่น ข้อความ, รูปภาพ, วิดิโอ และ เสียง เป็นต้น ซึ่งแต่ละชนิดนั้นจะมีโครงสร้างข้อมูล (Data Structure) ที่แตกต่างกัน อีกทั้งยังมีปริมาณที่เยอะขึ้น ทำให้เครื่องคอมพิวเตอร์เครื่องเดียวไม่สามารถทำงานได้อีกต่อไป

ดังนั้นจึงมีระบบที่เรียกว่า HADOOP ขึ้นมา เพื่อใช้ในการรวมพลังประมวลผลของคอมพิวเตอร์หลายๆ เครื่อง มาช่วยกันจัดการข้อมูลที่มีขนาดใหญ่ ซึ่งจะประกอบด้วย 2 ส่วนหลัก คือ HDFS และ YARN โดยมีรายละเอียด ดังนี้



1.1 HDFS (Hadoop Distributed File System) เป็นระบบจัดเก็บข้อมูลขนาดใหญ่ โดยลักษณะการเก็บจะเก็บในรูปแบบ Block โดยสมมติว่าเรามีข้อมูลขนาด 600MB ระบบ HDFS จะแบ่งข้อมูลเป็น 128MB เป็นจำนวน 4 ก้อน และ 88MB จะเก็บไว้ก้อนที่ 5 ดังตัวอย่าง



HDFS นั้นยังประกอบไปด้วย 2 ส่วนย่อย

(a.) NameNode (Master Node) คือ เป็นส่วนเก็บ Metadata หรือข้อมูลที่อธิบายรายละเอียดเกี่ยวกับข้อมูล เช่น จำนวนของบล็อก, เครื่องที่จัดเก็บ และข้อมูลอันไหนถูกเก็บไว้ที่ไหน เป็นต้น

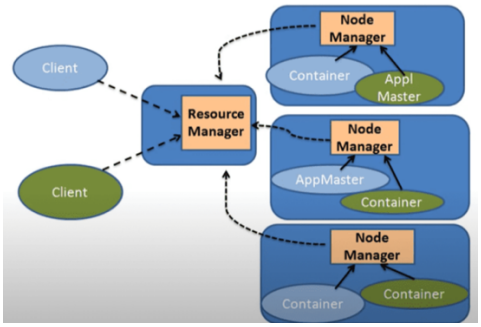
(b.) DataNode (Slave Node) คือ เป็นส่วนที่เก็บข้อมูลจริงของ HDFS ทำหน้าที่อ่านและเขียนตามคำสั่ง นอกจากนี้ยังทำหน้าที่ทำสำเนาข้อมูล (Replica) ของแต่ละบล็อกเก็บไว้ในอีก DataNode เพื่อป้องกันการสูญหายของข้อมูลอีกด้วย ทำให้เมื่อเกิดเหตุฉุกเฉินขึ้นมา ข้อมูลก็ยังอยู่กับเราครบ เรียกสถานการณ์แบบนี้ว่า มี Fault-Tolerant

1.2 YARN (Yet Another Resource Negotiator) เป็นส่วนสำคัญมากที่สุดส่วนหนึ่งใน ระบบ Hadoop เพราะทำหน้าที่จัดตารางเวลาให้งาน (Task) และบริหารจัดการทรัพยากร (Resource) ต่าง ๆ เช่น หน่วยเก็บข้อมูล และหน่วยประมวลผลบนระบบ ช่วยให้ทำงานได้ลื่นไหล เป็นเหมือนผู้คุมงานที่คอยจัดสรรงาน และคนงาน

YARN มีส่วนประกอบหลักๆตามด้านล่างนี้

- Client: คนที่ส่งงานที่ต้องทำออกไป
- Resource Manager: ทำการจัดการทรัพยากรที่มี และจำนวน Node ที่ยังสามารถทำงานได้
- Node Manager: คอยติดตามควบคุมงานของ Container รวมถึงแบ่งทรัพยากรในการคำนวณใน Container
- Container: เป็น package ที่มีทรัพยากรในการคำนวณอย่าง RAM, CPU และ Network
- Application Manager: ทำหน้าที่รันงานที่สั่งมา และคุยกับ Resource Manager เรื่องทรัพยากรในการรันงาน

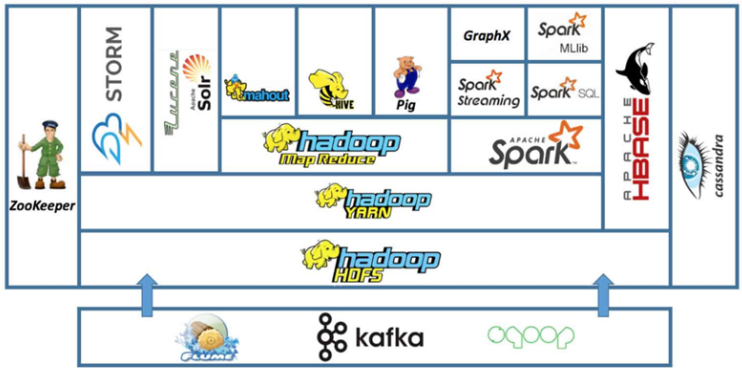
ซึ่งสามารถเห็นเป็นขั้นตอนการทำงานที่ชัดเจนได้ ตามภาพประกอบ ด้านล่างนี้



เมื่อ Client ต้องการทำงานอย่างหนึ่ง ระบบประมวลผลก็จะสั่งงานไปที่ Resource Manager หลังจาก Resource Manager เช็คความพร้อมแล้ว ก็จะติดต่อให้ Node Manager สร้าง Container และ รัน Application Manager ในนั้น ซึ่ง Application Manager จะทำการสั่งการรันงานโดยตรวจสอบ DataNode ใน HDFS แล้วแบ่งงานไปตาม Node จะมีการอัพเดทอยู่เรื่อย ๆ ว่าทำงานสำเร็จหรือไม่ ถ้าล้มเหลว Application Manager ก็จะเปลี่ยน Container แล้วลองรันงานอีกครั้ง

Hadoop Ecosystem

นอกจากนี้ยังมีส่วนประกอบอื่น ๆ อีกมากมายใน Ecosystem ที่สามารถนำมาใช้ร่วมกับ Hadoop ได้ทั้ง kafka (โปรแกรมในการจัดคิว), Apache Spark (ใช้งานได้ดีกับ Big Data), Cassandra (Database แบบ NoSQL), Hive (เครื่องมือในการดึงข้อมูลจาก Hadoop ผ่านทาง SQL) และ Pig (คล้ายๆกับ Hive ที่ทำให้ประมวลผลโดยไม่ต้องเขียน MapReduce แต่ใช้ภาษา Script) เป็นต้น



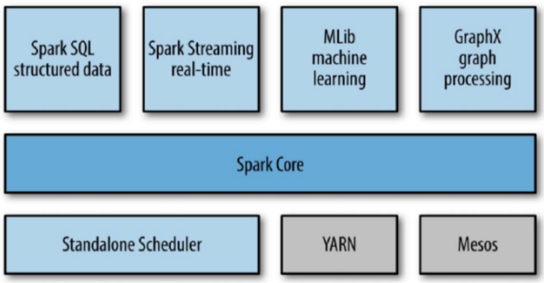
Hadoop Ecosystem

2. Apache Spark

Apache Spark เป็นเครื่องมือหนึ่งที่ถูกดีไซน์ให้สามารถทำงานแบบกลุ่มได้ โดยที่เชื่อมต่อระบบการทำงานของคอมพิวเตอร์เข้าด้วยกัน หรือเรียกว่า “Cluster computing platform” ซึ่งสามารถกระจายงานที่ต้องทำไปยังเครื่องอื่น ๆ ภายในระบบได้ ทำให้เราสามารถประมวลผลข้อมูลขนาดใหญ่แบบเต็มประสิทธิภาพ หรือแบบ real-time ไปพร้อม ๆ กันได้

จุดเด่นของ Apache Spark คือ ความเร็วที่ได้งาน เช่น มีงานหนึ่งที่ต้องใช้เวลา 1 ชั่วโมงในการทำงานด้วยคอมพิวเตอร์เครื่องเดียว แต่ถ้าเราใช้ Apache Spark จะสามารถแบ่งงานออกเป็นส่วนๆ ตามจำนวน node ที่ว่างเพื่อช่วยในการทำงานให้เสร็จเร็วขึ้นได้

จุดเด่นอีกอย่างหนึ่งของ Apache Spark คือ สามารถใช้งานทั่วไปได้ ทำให้โครงสร้าง Spark ประกอบไปด้วยหลายส่วนด้วยกัน คล้ายกับการใช้ Libraries ในซอร์ฟแวร์อื่นๆ เพื่อรองรับกับงานที่หลากหลายได้ ตั้งแต่ SQL จนถึง Machine Learning เรียกได้ว่าครบวงจรเลยก็ว่าได้

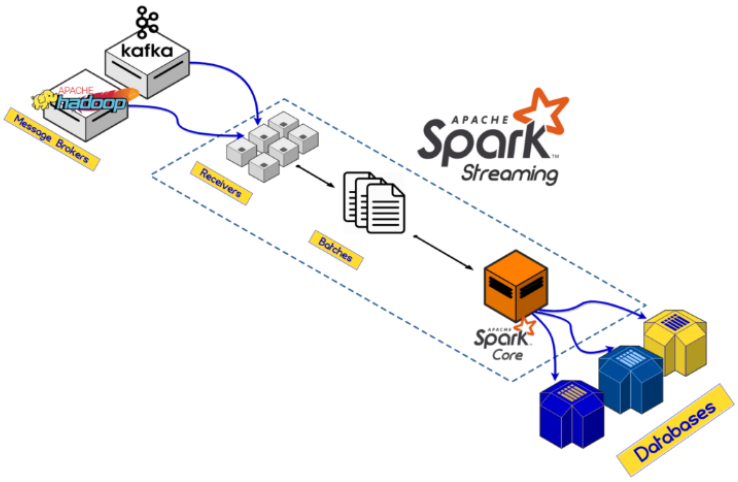


ส่วนประกอบหลักของ Spark

- *Spark Core:* มีฟังก์ชั่นเบื้องต้นสำหรับ กำหนดงาน จัดการหน่วยความจำ Fault Recovery การจัดการ Storage System และอื่นๆอีก เช่น การเรียก API เฉพาะตัว ที่เรียกว่า RDDs (Resilient Distributed Datasets) ซึ่งเป็นสิ่งที่ช่วยเรื่องการรันโปรแกรมในหลายๆเครื่อง พร้อมๆกันแบบคู่ขนาน

- *Spark SQL:* เป็นส่วนที่สามารถใช้ SQL, HQL จัดการกับข้อมูลที่มีโครงสร้างในรูปแบบต่างๆได้ เช่น Hive table, Parquet และ JSON เป็นต้น นอกจากนี้ยังมีความพิเศษตรงที่เราสามารถใช้ SQL queries กับภาษาอื่นๆ เช่น Python, Java, Scala ผ่าน RDDs ได้ในที่เดียว
- *Spark Streaming:* เป็นส่วนที่ทำให้เราทำงานกับดาต้า แบบ Live Streams ได้ ซึ่งมีการใช้ API ในการจัดการที่คล้ายกับ RDDs ทำให้มันง่ายกับโปรแกรมเมอร์ในการสลับไปมา ระหว่างดาต้าใน หน่วยความจำ, ดิสก์ หรือ ดาต้าที่มาถึงแบบ Real Time
- *MLlib:* มีฟังชั่นในการทำ Machine Learning ตั้งแต่เบื้องต้นจนถึงแอดวานซ์ ยกตัวอย่างเช่น Classification, Regression, Clustering และ Collaborative Filtering รวมถึงการประเมินโมเดล และการนำข้อมูลเข้ามาใช้วิเคราะห์
- *GraphX:* เป็นส่วนที่ใช้จัดการกราฟ เช่น กราฟที่เชื่อมความสัมพันธ์ระหว่างเรากับเพื่อนใน Social Media สามารถใช้ RDDs ในการควบคุมแต่ละ Vertex และ Edge ของกราฟได้โดยตรง อีกทั้งยังมี Operation และ Libraries ต่างๆเข้ามาเสริมด้วย
- *Cluster Managers:* อย่างที่รู้กันดีว่า Spark ทำให้เราทำงานพร้อมกันในหลายเครื่อง (nodes) ได้ เพื่อให้ง่ายต่อการทำงานในการแบ่งคลัสเตอร์จึงมี เครื่องมือในการจัดการคลัสเตอร์ในตัวเอง เรียกว่า “Standalone Scheduler” แต่สำหรับใครที่มีเครื่องมืออื่นอยู่แล้ว เช่น Hadoop YARN, Apchae Mesos ก็สามารถใช้งานด้วยกันได้

โดยสรุปแล้วจากเหตุผลข้างต้นทางทีมเลือกใช้ Hadoop และ Apache Spark ถูกนำมาเป็นตัวช่วยในการประมวลผลข้อมูล Cryptocurrency สำหรับทำ Automate Trading Bot โดย Hadoop นั้นถูกนำมาเป็นระบบจัดเก็บข้อมูลขนาดใหญ่ และ Spark นั้นถูกนำมาใช้เป็นตัวประมวลผลแบบคู่ขนาน โดยการใช้สองเครื่องมือนี้ร่วมกันเพิ่มประสิทธิภาพทั้งในแง่ของการประมวลผลและลดค่าใช้จ่ายในการรันโปรแกรม ตั้งแต่การ Deployment, Maintenance, Testing หรือ Support อีกด้วย



การจัดเก็บข้อมูล (Data Store)

Data Store ในงานวิจัยครั้งนี้ หลังจากข้อมูลได้รับการประมวลผลเรียบร้อยแล้ว ก็จะถูกจัดเก็บไว้ใน Database เพื่อที่จะเตรียมนำไปใช้ประมวลผลต่อ โดยในการเลือก Database นั้นก็มีหลายเทคโนโลยีให้เลือก ทางทีมได้เลือกเปรียบเทียบ Database หลายแบบโดยมีข้อมูลดังนี้

1. Relational Database

โดย Relational Database ที่ทางทีมได้ศึกษาได้แก่ PostgreSQL โดยมีจุดเด่น-จุดด้อยดังนี้

จุดเด่น:

- PostgreSQL เป็น open-source object-relational system ที่ถูกใช้งานในรูปแบบเดียวกับภาษา SQL ซึ่ง PostgreSQL จะช่วยให้คุณเก็บข้อมูลไว้ในฐานข้อมูลขนาดใหญ่อย่างปลอดภัย และมันยังช่วยสร้างแอพพลิเคชั่นที่ซับซ้อน, รันจัดการ Task และสร้าง environmental ที่สำคัญได้ด้วย
- PostgreSQL สามารถจัดการ compelling request ของบริษัท และสถาบันขนาดใหญ่ ได้มากที่สุด ซึ่งประสิทธิภาพของมันถูกพิสูจน์มาแล้วกว่าหลายปี หรืออีกแง่หนึ่ง PostgreSQL ไม่เคยหยุดพัฒนา และยังคงปล่อยเวอร์ชั่นใหม่มาเรื่อย ๆ หนึ่งในนั้นคือการพัฒนาเพื่อรองรับ ข้อมูลประเภท unstructured ตัวอย่างที่ดังสุด ก็คือ JSONB feature.
- PostgreSQL ให้เครื่องมือ และ Featureต่างๆ ในการตรวจสอบความถูกต้องของข้อมูล Database โดยจะพิจารณาตรวจสอบความถูกต้องของข้อมูลสำหรับ JSON field ดังนั้นหากมีการเข้ามาไม่ถูกต้อง มันจะถูกแจ้ง Error ทันที
- ภาษาโปรแกรมสำหรับเข้า Server ของ PostgreSQLนั้น รองรับการจัดการด้วยภาษาที่หลากหลาย อาทิ Python, JavaScript, C, C++, Tcl, Perl และอื่นๆอีกมากมาย
- PostgreSQLนั้น สามารถทำงานสอดประสานได้ดีกว่าคู่แข่ง
- PostgreSQLนั้น สามารถรองรับการขยายตัวในอนาคตได้ มันซัพพอร์ต ประเภทข้อมูลได้หลากหลาย อาทิ geometric/GIS, network address types, JSONB, native UUID, timezone-aware timestamps
- PostgreSQL นั้นถูกใช้แพร่หลาย สำหรับกลุ่มลูกค้าแอพลิเคชั่นรายย่อย และ ขนาดกลาง ด้วยลูกเล่นมากมายและประสิทธิภาพในการเก็บข้อมูล

จุดด้อย:

- ประสิทธิภาพในระดับเมตริก และความเร็วอาจไม่ไวมาก
- การบำรุงรักษาและซ่อมแซมระบบที่เกี่ยวข้องกับ RDBMS นั้นต้องใช้ความพยายามและการทำงานมากกว่า

2. NoSQL Database

นอกจากนี้ทางทีมยังได้ศึกษา NoSQL database เช่นกัน ได้แก่ MongoDB ซึ่งก็คือ document based database โดยมีจุดเด่น-จุดด้อยดังนี้

จุดเด่น:

- MongoDB เป็น database แบบ Document-Oriented โดยลักษณะการเก็บข้อมูลจะใช้รูปแบบ format เป็น Json Style โดย Row แต่ละ Row ไม่จำเป็นต้องมีโครงสร้างข้อมูลเหมือนกัน
- MongoDB ใช้ระบบการจัดการ memory แบบเดียว กับ cached memory ใน Linux ซึ่งจะปล่อยให้ OS เป็นคนจัดการ Memory
- ใช้ภาษา JavaScript เป็นคำสั่งในการจัดการข้อมูล
- MongoDB เป็น Full Index กล่าวคือรองรับข้อมูลมหาศาลมากๆ และสามารถค้นหาจากส่วนไหนของข้อมูลเลยก็ได้
- MongoDB รองรับการ เพิ่ม หรือ หด field แบบรวดเร็ว โดยไม่ต้องใช้คำสั่ง Alter Table

จุดด้อย:

- MongoDB นั้นไม่ซัพพอร์ตการทำธุรกรรม ซึ่งถ้า application ใด ต้องการฟังชั่นนี้ ทาง MongoDB ก็อาจไม่เหมาะ
- การ Join ข้อมูลใน MongoDB นั้น ไม่ง่ายนัก แม้ว่า จะมีการใช้ left-outer joins, developer ยังคงต้องทำงานบนฟังชั่นนี้ และมันก็ยังไม่สมบูรณ์พอ การใส่ข้อมูลบางส่วนจาก collections ที่ต้องการจำนวนการ queries ซึ่งเลี่ยงไม่ได้ที่ต้องเจอกับ code แปลกๆ และการรันที่ใช้เวลานาน
- ด้วยการดึง indexing อย่างหยาบๆ และ ไม่มีการกำหนดองค์ประกอบของการดึง ทำให้ MongoDB จัดการเรื่องนี้ได้ช้ามาก
- กินพื้นที่การเก็บข้อมูลมากกว่า MySQL พอสมควร เพราะไม่มี Schema ดังนั้น Schema จริงๆจะอยู่ในทุก row ของฐานข้อมูล ทำให้ข้อมูลใหญ่กว่า MySQL
- หากใช้งานจน disk เต็ม จะ clear พื้นที่ disk ให้ใช้งานต่อยาก เพราะการสั่ง delete row ไม่ทำให้ฐานข้อมูลเล็กลง ต้องสั่ง compact เองซึ่งต้องมีพื้นที่ว่างของ disk อีกลูกมากพอๆ กับพื้นที่ข้อมูลที่ใช้อยู่ปัจจุบันเป็น buffer ในการลดขนาด
- ความสัมพันธ์ ใน MongoDB อาจไม่ได้ระบุชัดเจนนัก และการ Duplicate อาจยากต่อการ handle ซึ่งมันขาดคุณสมบัติ ACID (Atomicity, Consistency, Isolation และ Durability) ทำให้อาจถูก corrupt ได้ง่าย

หลังจากที่ได้เปรียบเทียบ Data Store ทั้ง 2 แบบนี้แล้ว ทางทีมได้ลงความเห็นว่า Relational Database (PostgreSQL) คือตัวเลือกที่เหมาะสมที่สุด ด้วยเหตุผลหลักๆ ดังนี้

- PostgreSQL เป็น open-source object-relational system ที่ถูกใช้งานในรูปแบบเดียว กับภาษา SQL ซึ่งมีความ Strong ในการ Queries และรองรับ Data Type ทุกประเภท อาทิ documents, primitives, geometry, structures เป็นต้น
- ในการแปลงข้อมูล (Data integrity) จากหลากหลายช่องทาง PostgreSQL จะพิจารณา การ integrate ข้อมูลของคุณ ด้วยการแนะนำข้อจำกัด และ ควบคุมข้อมูลที่คุณใส่เข้าไป ด้วย PostgreSQL คุณสามารถลืมเรื่อง invalid หรือ orphan record ไปได้เลย
- มี Performance: ในการทำงานคู่ขนานในการ queries, การ indexing methods ที่มีพลัง และการควบคุมสอดประสานหลายเวอร์ชั่นพร้อมกัน
- รองรับภาษา Python ในการเข้า Server ของ PostgreSQL ซึ่งเป็นภาษาหลักที่ใช้ในการทำ Data pipeline ของโครงการนี้ด้วย
- Extensibility: การขยายขนาด ในฐานข้อมูลนี้ คุณไม่ต้องจำกัดตัวเองว่า ต้องใช้ข้อมูลประเภทใดประเภทหนึ่งเท่านั้น เพราะ ฐานข้อมูลนี้ ให้คุณเลือกเก็บข้อมูลได้หลากหลาย เพื่อความไม่มีลิมิตของคุณ
- Disaster Recovery & Reliability: การกู้คืน และความน่าเชื่อถือ PostgreSQL ใส่ใจการรักษาระดับความน่าเชื่อถือให้กับข้อมูลของคุณ ด้วยออฟชั่น replication (การทำสำเนา) ข้อมูลของคุณจึงปลอดภัยหายห่วง ในอีกแง่หนึ่ง คุณสามารถ Backup ข้อมูลที่มีคุณค่าของคุณได้เสมอ
- Support of non-relational data: ซัพพอร์จข้อมูลที่เป็น non-relational, ซึ่งมันสำคัญมากในการอัพเดตฐานข้อมูล ซัพพอร์ตพวกไฟล์ JSON, XML, Hstore และ Cstore documents โดยแปลงจาก Postgres ไปเป็น NoSQL database

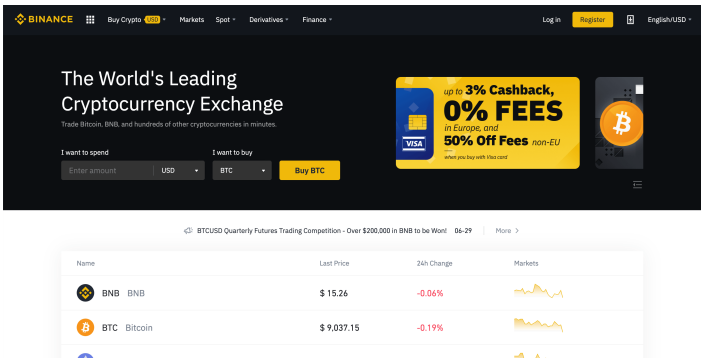
กลยุทธ์การซื้อ-ขาย

หลังจากข้อมูลถูกนำ ไปเก็บไว้ใน Database เรียบร้อย ข้อมูลที่พร้อมถูกใช้งานก็จะถูกนำมาประมวลผลเพื่อคาดการณ์และค้นหาคำสั่งซื้อ-ขาย Cryptocurrency โดยทางทีมได้ใช้เครื่องมือที่เรียกว่า CDC Action Zone Version3 มาเป็น Indicator ในการหาคำสั่งซื้อขายโดย CDC Action Zone เป็นการคำนวณหาค่า exponential moving averages (EMA) สองค่าได้แก่ EMA 12 วัน และ 26 วัน แล้วนำค่าที่ได้มาเปรียบเทียบเพื่อหาสัญญาณซื้อ หรือสัญญาณขาย โดยค่า EMA สามารถคำนวณได้จากสมการด้านล่างนี้

$$\text{Exponential Moving Average} = C - P \times \frac{2}{(n+1)} + P$$

โดยสัญญาณซื้อจะเกิดขึ้นก็ต่อเมื่อค่า EMA 12 > EMV 26 เป็นครั้งแรกเมื่อเทียบกับช่วงเวลาที่ผ่านมา ในขณะที่สัญญาณขายจะเกิดขึ้นเมื่อ EMA 12 < EMV 26 เป็นครั้งแรกเมื่อเทียบกับช่วงเวลาที่ผ่านมา จากรูปด้านล่างเป็นตัวอย่างของสัญญาณซื้อและสัญญาณขายที่เกิดขึ้น โดยจะเกิดขึ้นเมื่อค่า EMA 12 และ EMA 26 วันมีการตัดขึ้น หรือตัดลง โดยจะบ่งบอกถึงสัญญาณซื้อและสัญญาณขายตามลำดับ

การส่งคำสั่งซื้อ-ขาย



หลังจากที่เราได้รู้แล้วว่าจะหาจังหวะในการซื้อ-ขาย Cryptocurrency ได้อย่างไรอีกหนึ่งสิ่งสำคัญที่ขาดไม่ได้เลยคือการที่เราจะสามารถส่งคำสั่งซื้อ-ขายไปได้อย่างอัตโนมัติ โดยในโครงการค้นคว้านี้ทางทีมได้เลือกใช้ Binance API ซึ่ง Binance เป็นผู้ให้บริการตลาดซื้อขาย Cryptocurrency ที่ให้ผู้ใช้บริการสามารถส่งคำสั่งการบริการจัดการพอร์ตการลงทุนของตัวเองผ่าน Restful API โดยคำสั่งจะประกอบไปด้วย การบริการส่งคำสั่งซื้อ-ขาย Cryptocurrency ในรูปแบบต่างๆ เช่น Limit Order, Market Order และสามารถเรียกดูจำนวน รวมถึงมูลค่าของเงินสินทรัพย์และ Cryptocurrency ที่มีอยู่ในพอร์ตการลงทุนของผู้ใช้งาน เป็นต้น

ในแง่ของการใช้งาน Binance API มีคู่มือในการใช้งานให้ครบถ้วน รวมถึงมีแผนกที่คอย Support ผู้ใช้งานกรณีเมื่อเกิดปัญหาขึ้นและสำหรับการพัฒนาแอพพลิเคชัน การใช้งานผ่าน Restful API นั้นอำนวยความสะดวกแก่ผู้ใช้งานโดยสามารถส่งคำสั่งซื้อขายโดยไม่จำเป็นที่ต้องเปิดเว็บไซต์หรือแอพพลิเคชั่นเพื่อส่งคำสั่งแต่สามารถเขียนโปรแกรมเพื่อส่งคำสั่งผ่าน API ได้เลยซึ่งเป็นบริการที่เหมาะกับการทำ Automate Trading Bot มากๆ

ข้อดี:

- เข้าถึงตลาดซื้อ-ขายที่มีสภาพคล่องสูง
- สามารถส่งคำสั่งซื้อ-ขายผ่าน Restful API ได้
- ไม่เสียค่าใช้จ่าย
- สามารถยกเลิกคำสั่งซื้อ-ขายได้ผ่าน API

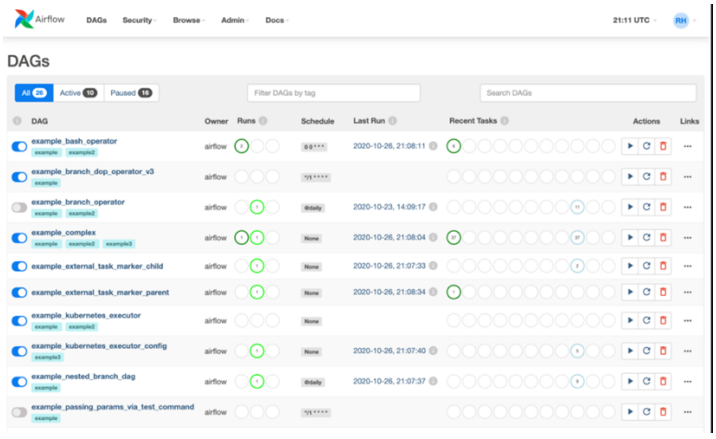
ข้อเสีย:

- ไม่สามารถส่ง API Request ได้มากกว่า 1,200 คำสั่งต่อนาที
- หากส่งคำสั่งเกินจำนวนที่กำหนด จะถูกระงับบัญชีชั่วคราว

การสร้าง Data Pipeline

ในการทำงานวิจัยครั้งนี้ทางกลุ่มเลือกใช้ Apache Airflow มาเป็นตัวจัดการ Data Pipeline โดย Apache Airflow คือ หนึ่งใน Workflow Management ที่ได้รับความนิยมอย่างมาก และองค์กรชั้นนำระดับโลกหลายๆ องค์กรต่างเลือกใช้ โดยเฉพาะอย่างยิ่งในการสร้าง Data Pipelines เพื่อจัดการกับข้อมูลจำนวนมหาศาลส่วนหนึ่งเพราะองค์กรต่าง ๆ ในปัจจุบัน ต่างพยายามเปลี่ยนแปลงให้องค์กรขับเคลื่อนด้วยข้อมูลจำนวนมหาศาล (Data-Driven) แต่กว่า 73% ของข้อมูลในองค์กรกลับไม่ได้ถูกนำมาวิเคราะห์เลย ซึ่งเหตุผลหลักๆ ก็คือข้อมูลไม่ได้ถูกจัดการให้เป็นระเบียบนั่นเอง นอกจากนี้ องค์กรเป็นจำนวนมากก็ยังเก็บข้อมูลที่ไร้ประโยชน์ ไม่ว่าจะเป็น Web Log, อีเมลเก่าๆ, หรือข้อมูลลูกค้าที่ Out of date และเก็บมาแล้วไม่ได้ใช้ ไว้เป็นจำนวนมาก ในขณะที่ข้อมูลเพิ่มขึ้นเรื่อย ๆ หลาย ๆ อย่างก็เริ่มเข้ามาเป็นข้อจำกัด เช่น กฎหมาย PDPA เป็นต้น ทำให้การจัดการข้อมูลเข้ามามีบทบาทสำคัญมากในองค์กรยุคปัจจุบัน

Airflow คือ เครื่องมือหนึ่งในการสร้าง Data Pipeline โดยเป็น Open Source Platform ตัวหนึ่งที่ผู้ใช้สามารถเขียนโปรแกรมที่จะมาควบคุมการไหลของ Workflow และสามารถคอยเฝ้าดูการไหลได้อีกด้วย ซึ่งถูกพัฒนาโดย Airbnb และเริ่มใช้งานมาตั้งแต่ปี 2015 ซึ่งนอกจากสร้าง Data Pipelines ได้แล้ว มันยังสามารถนำไปสร้าง หรือ พัฒนา ETL (Extract-Transform-Load), Machine Learning และ Predictive ได้อีกด้วย โดย Airflow เข้ามาช่วยจัดการ Task ต่าง ๆ อีกทั้งยังรองรับแบบ Hybrid & Multi-Cloud โดยเกิดขึ้นมาจาก Data Pipeline ที่มีจำนวนมากซึ่งทำให้ควบคุมได้ยาก โดย Airflow มีการเขียน Workflow เป็น DAG (Directed Acyclic Graph) กราฟที่มีหัวลูกศรทิศทางเดียว โดยไม่สามารถวนกลับมาที่จุดเดิมได้ ซึ่ง DAG ประกอบไปด้วยหลายๆ Task ที่เชื่อมต่อกัน และในแต่ละ Task นั้นก็มีความสามารถที่แตกต่างกัน (ดังภาพประกอบด้านล่าง)



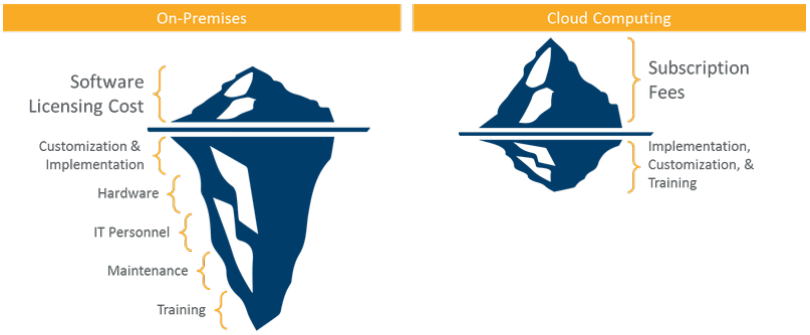
หลักการของ Apache Airflow (Airflow Principles)

- *Scalable* โดยเราสามารถปรับแต่ง Airflow ให้สามารถรับมือกับข้อมูลจำนวนมหาศาลได้
- *Dynamic* โดยเราสามารถเขียนโค้ดเพื่อ Generate Pipeline หรือ Workflow ขึ้นมาได้
- *Extensible Airflow* อนุญาตให้เราเขียน Operators ใหม่ๆ ขึ้นมาได้จาก Core Operators เดิมเพื่อให้ตอบโจทย์ตามความต้องการของผู้ใช้งานได้
- *Elegant* เราสามารถส่งค่า หรือข้อมูลอะไรบางอย่างเข้าไปใน Pipeline ได้แม้ว่าตัว Data Pipelines ของเรากำลังทำงานอยู่

เหตุผลที่ทางทีมเลือกใช้ Apache Airflow

- *User Interface:* โดยตัวระบบมีหน้าตาการใช้งานที่ค่อนข้างทันสมัย และมีความเป็น User Friendly สูงทำให้เข้าใจส่วนต่างๆ ได้ง่าย
- *Python:* เราสามารถเขียนด้วยภาษา Python เพื่อกำหนดงานต่าง ๆ ใน Data Pipelines ได้ ทำให้ง่ายต่อการเริ่มต้นเรียนรู้ และใช้งาน
- *General purpose:* เราสามารถใช้สร้าง Data Pipelines แบบไหนก็ได้ หรือเอาไปใช้เพื่อทำ Automate อื่น ๆ ได้ โดยไม่ต้องทำเพียงแค่ Data Pipeline อย่างเดียว
- *Easy to add new functionality:* ง่ายต่อการเพิ่มฟังก์ชั่นใหม่ๆ เนื่องจากถูกพัฒนามาเพื่อให้ผู้ใช้สามารถพัฒนาต่อยอดได้ง่ายจากความสามารถเดิม
- *Easy to monitor:* สามารถติดตามผลการทำงานของ Workflow ใน State ต่าง ๆ ได้ง่ายผ่านหน้าตาของ UI ที่มีได้เลย
- *Scale:* ระบบถูกออกแบบมาให้ขยายขอบเขตการทำงานได้ง่ายเพื่อการ Scale ในอนาคต
- *Community:* มี Community ของผู้ใช้งานที่ใหญ่ และ ใน Github ก็มีตัวอย่างโปรเจคมากมาย ซึ่งการมี Community ขนาดใหญ่แบบนี้จะช่วยให้เวลาที่ผู้ใช้ติดปัญหาการใช้งาน จะสามารถแก้ปัญหาในระบบได้อย่างรวดเร็ว

On-Cloud vs On-Premise



สำหรับการประมวลผล ทางทีมได้เปรียบเทียบข้อแตกต่างระหว่าง On-Cloud vs On-Premise แล้วเห็นพ้องกันว่า On-Cloud Service นั้นมีข้อดีที่น่าสนใจ และสอดคล้องกับการทำ Data pipeline ดังนี้

1. *ประหยัด:* เพราะไม่ต้องเสียเงินที่เป็น Initial Cost ในการลงทุนสร้างระบบ รวมถึงการ Service ในฝั่งของฝ่าย IT ที่เป็น Ongoing Cost
2. *ครอบคลุมทั่วโลก:* การใช้บริการ Cloud Computing เราสามารถเลือกที่ตั้งของ Data Center ได้ เช่น จะให้อยู่ในแถบเอเชีย อเมริกา หรือยุโรปก็ได้ โดยเราไม่จำเป็นต้องไปสร้าง Data Center ด้วยตัวเองในประเทศต่างๆ
3. *ประสิทธิภาพ:* เนื่องจากบริการนี้ให้บริการอยู่ทั่วโลก ดั้งนั้นผู้ให้บริการ ไม่ว่าจะเป็นแบรนด์ไหนก็แล้วแต่ ต้องอัพเกรดระบบอยู่อย่างเสมอ เพื่อให้มีประสิทธิภาพสูงสุด เพื่อรองรับการให้บริการทั่วโลก ต่างกับการที่คุณมี Local ของตัวเอง การอัพเดทก็อาจช้า ไม่ทันเทคโนโลยีใหม่ๆ ระยะยาวอาจทำให้ประสิทธิภาพลดลงได้ และสิ้นเปลืองค่าใช้จ่ายมากกว่าด้วย
4. *ความเสถียรในการใช้งาน:* ระบบจะมีคำสั่งในการ Backup ข้อมูล หรือ กู้ฐานข้อมูลกลับ (Data Recovery) ได้ง่ายกว่า ช่วยให้ฝ่าย IT ดูแลระบบได้ง่ายขึ้น และยังมีระบบด้านความปลอดภัยต่างๆ ที่ช่วยส่งเสริมด้าน Cyber Security และป้องกันการโจมตีทางไซเบอร์

โดยเราขอเปรียบเทียบคุณสมบัติของ 3 บริษัท ได้แก่ AWS (Amazon Web Server), Microsoft Azure และ GCP (Google Cloud Platform) ดังนี้

AWS (Amazon Web Server)

แพลตฟอร์มผู้ให้บริการ Cloud Computing มาเป็นเวลานาน ด้วย บริการที่หลากหลายและครอบคลุมกว่า 175 บริการ เช่น ด้าน Analytic, Blockchain, Game, Internet of Things, Machine Learning, Quantum Technology ทำให้สามารถตอบโจทย์ผู้ใช้ระดับองค์กรได้เป็นอย่างดี

1. ฟีเจอร์ที่ให้บริการ

- บริการ The instances of virtual servers และ Virtual machines.
- ESC สำหรับการจัดการ Docker management
- Glacier archive storage
- บริการ Amazon cloud search
- วิเคราะห์ข้อมูลโดย Amazon Kinesis
- AWS ops work and config สำหรับบริการระบบอัตโนมัติ

2. ด้านราคาและค่าบริการ สามารถซื้อได้ตามแบบ On-demand เลือกบริการและคิดค่าใช้จ่ายตามจริง หรือคิดค่าบริการเป็นรายชั่วโมง

3. ด้านความปลอดภัย มีฟีเจอร์ AWS cloud HSM และ AWS key management service สำหรับข้อมูลความปลอดภัย

Microsoft Azure

บริการ Cloud Computing จาก Microsoft ซึ่งมีฟังก์ชันและฟีเจอร์ย่อยหลากหลาย เรียกได้ว่าเป็นบริการ Cloud Server ที่ครบครันเจ้าหนึ่ง ทั้งเครื่องมือด้าน AI, Machine Learning, Blockchain, DevOps, Internet of Things, Storage, Web และบริการอื่น ๆ อีกมากมาย

1. ฟีเจอร์ที่ให้บริการ มีตั้งแต่บริการ The virtual hard disks ที่ช่วยสร้าง Instance ได้ครบครัน รวมถึงต่อยอดการทำงานได้อย่างหลากหลาย เช่น

- Container service สำหรับการจัดการ Docker management
- บริการ Azure arc Container service สำหรับการจัดการ Docker management
- Azure archive storage
- บริการ Azure search
- วิเคราะห์ข้อมูลโดย Azure stream analytics
- Azure automation สำหรับบริการระบบอัตโนมัติ

2. ราคาและค่าบริการ สามารถซื้อได้ตามแบบ On-demand แบบระยะสั้น (short termed commitments) และคิดค่าบริการเป็นรายนาที

3. ด้านความปลอดภัย มีบริการ Azure trust center และ Azure key vault สำหรับข้อมูลความปลอดภัย

Google Cloud Platform

ผู้ให้บริการ Cloud Server จาก Google ที่เราคุ้นเคยกันเป็นอย่างดี ซึ่งมีฟีเจอร์การทำงานที่หลากหลาย เนื่องจากเป็นบริษัทขนาดใหญ่อยู่แล้ว จึงทำให้มีบริการที่หลากหลายไม่แพ้ Azure หรือ AWS

1. ฟังก์ชันหรือฟีเจอร์ที่ให้บริการ มีบริการพื้นฐาน The virtual machine instances และ Google container engine สำหรับการจัดการ Docker management นอกจากนี้ยังมีฟีเจอร์อื่น ๆ ที่หลากหลาย เช่น

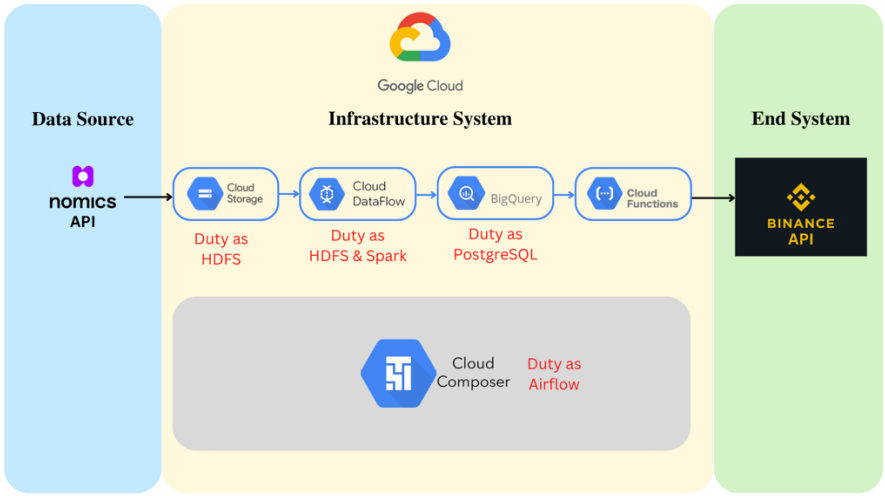
- บริการ Cold line archive storage
- ไม่มี search service
- วิเคราะห์ข้อมูลโดย Cloud dataflow และ Cloud data prepare
- บริการ google cloud และ compute engine management พร้อมด้วย the puppet, chef และอีกมากมาย สำหรับบริการระบบอัตโนมัติ

2. ด้านราคาและค่าบริการ สามารถซื้อได้ตามแบบ On-demand แบบระยะยาวใช้งานต่อเนื่อง (sustain use) และคิดค่าบริการเป็นรายนาที

3. ด้านความปลอดภัย มีฟีเจอร์ Google's cloud platform security สำหรับข้อมูลความปลอดภัย

หลังจากที่เปรียบเทียบทั้ง 3 บริษัทแล้ว ทางทีมได้ตัดสินใจเลือกใช้บริการ Google Cloud Platform ที่มีบริการที่ชื่อว่า "Google Compute Engine" หรือเรียกย่อ ๆ ว่า GCE ซึ่งเป็นบริการเครื่องคอมพิวเตอร์เซิร์ฟเวอร์ให้กับลูกค้าที่ต้องการนำไปทำงานต่าง ๆ เป็นระบบที่เราสามารถสร้าง เครื่องคอมพิวเตอร์เซิร์ฟเวอร์ได้อย่างรวดเร็ว ลดเวลาจากสัปดาห์มาเหลือแค่หลักนาทีเท่านั้น ซึ่งบริการนี้ทาง Google Cloud Platform มีสเปคของเครื่องคอมพิวเตอร์เซิร์ฟเวอร์ที่หลากหลาย เราสามารถสร้างเองได้ตามความเหมาะสมกับประเภทของงานที่ใช้ ไม่ว่าจะเป็นสเปคของ ซีพียู ขนาดของหน่วยความจำ หรือ ที่เก็บข้อมูล รวมถึงกำหนดระบบปฏิบัติการที่ต้องการใช้ได้ อาทิ เช่น Windows หรือ Linux เป็นต้น

Final Data Pipeline



จากที่นำเสนอมาทั้งหมดนำมาสู่โครงสร้างของ Data Pipeline สำหรับทำ Automate Trading Bot ในโครงการค้นคว้านี้ โดยโครงสร้างสามารถแบ่งเป็นสามส่วนหลักๆได้แก่

1. *Data Source* ได้แก่ที่มาของข้อมูลราคา Cryptocurrencyในการนำมาสร้าง Automate Trading Bot ในโครงการค้นคว้านี้ได้ดึงจาก Nomics API ผ่าน Restful API
2. *Infrastructure System* คือส่วนหลักในการจัดการข้อมูลทั้งในส่วนการนำข้อมูลมาประมวลมวลผลและจัดเก็บแบบ ETL โดยทางทีมได้ใช้ Service ต่างๆของ Google Cloud Platform1 โดยสามารถแบ่งได้ดังนี้
   - Cloud Storage ทำหน้าที่เป็น Hadoop (HDFS) เพื่อจัดการข้อมูลขนาดใหญ่
   - Cloud Data Flow ทำหน้าที่เป็น Apache Spark เพื่อจัดการ การประมวลผลข้อมูลแบบคู่ขนาน
   - Big Query ทำหน้าที่เป็น PostgreSQL เพื่อจัดเก็บข้อมูลที่ถูกประมวลผลแล้ว
   - Cloud Function ทำหน้าที่เป็นตัวประมวลผลหาสัญญาซื้อขายและทำหน้าที่ส่งคำสั่งซื้อขายไปยัง Binance API
   - Cloud Composer ซึ่งได้แก่ Apache Airflow ทำหน้าที่เป็นตัวจัดการ Data Pipeline
3. *End System* ได้แก่ Binance API ซึ่งทำหน้าที่รับคำสั่งซื้อขาย Cryptocurrency

จากโครงสร้างข้างต้น จะเห็นได้ว่าโครงสร้างที่ทางทีมได้นำเสนอนั้น สามารถตอบโจทย์คำถามที่ว่าถ้าเราจะสร้าง Automate Trading Bot จะต้องแก้ปัญหาและมีส่วนประกอบอะไรบ้าง ดังชุดคำถามต่อไปนี้

1. จะใช้ภาษาอะไรในการทำ Data Pipeline?
2. ข้อมูลราคาของ Cryptocurrencyจะนำมาจากแหล่งไหน?
3. จะใช้ตัวช่วยอะไรในการจัดการและประมวลผลข้อมูล?
4. การจัดเก็บข้อมูลอย่างไร?
5. กลยุทธ์ของการซื้อ-ขายคืออะไร?
6. จะส่งคำสั่งซื้อ-ขายอย่างไร?
7. เครื่องมือที่ช่วยในการจัดการ Data Pipeline คืออะไร?
8. ตัวช่วยประมวลผลการดำเนินการควรเป็นอะไร ระหว่าง On-Cloud vs On-Premise?

และจะเห็นได้ว่าโครงสร้างนี้รอบรับการปรับเปลี่ยนและกลยุทธ์ในการซื้อ-ขายที่หลากหลายโดยสามารถปรับเปลี่ยนได้ในส่วนของ Cloud Function รวมถึงรองรับข้อมูลที่มีความถี่สูงเช่น Intraday เป็นต้นเพราะตัว Cloud Storage (HDFS) และ Cloud DataFlow นั้นสามารถรองรับและประมวลผลข้อมูลที่มีขนาดใหญ่ได้และนี่ก็เป็นโครงสร้างที่สามารถนำไปต่อยอดในทางอื่นๆได้อีกด้วยไม่ใช่แค่ในการสร้าง Automate Trading Bot แต่สามารถนำไปทำเป็น Dashboard ประมวลผลได้เช่นกันโดยการ Replace ส่วน End System ด้วย Visualization Tool กล่าวโดยสรุปโครงสร้างที่ทางทีมนำเสนอสามารถเป็นจุดเริ่มต้นในการนำไปปรับใช้เพื่อช่วยนักลงทุนในการจัดการข้อมูล Cryptocurrency ได้อย่างมีประสิทธิภาพมากยิ่งขึ้น

แหล่งเอกสารอ้างอิง

Cryptocurrency Data Pipeline

https://medium.com/bakhu-publication/how-to-create-a-data-pipeline-to-analyze-cryptocurrency-from-nomics-api-using-hadoop-spark-fcc87dbe09bd

https://systamental.com/building-crypto-data-pypelines-3b4c9136d301

https://towardsdatascience.com/obtaining-historical-and-real-time-crypto-data-with-very-simple-web-programming-7b481f153630

https://www.databricks.com/blog/2022/05/02/introduction-to-analyzing-crypto-data-using-databricks.html

Programming language

https://www.thedataincubator.com/blog/2022/08/08/10-technologies-to-build-your-data-pipeline/

https://www.infoworld.com/article/3049672/which-freaking-big-data-programming-language-should-i-use.html

https://www.g2.com/articles/big-data-programming-languages

https://blog.subnetzero.io/post/2018/11/grimwhisker-language-and-stack/

Data Source (API)

https://nomics.com/docs/#tag/Currencies-Ticker

https://www.abstractapi.com/guides/best-crypto-currency-apis

https://www.analyticsinsight.net/top-10-crypto-exchange-apis-to-look-for-in-2022/

https://nordicapis.com/9-apis-for-real-time-cryptocurrency-data/

https://nomics.com/blog/essays/cryptocurrency-market-data-apis

HADOOP

https://blog.datath.com/hadoop-hdfs-yarn-mapreduce/

https://drtan.medium.com/%E0%B8%AA%E0%B8%B2%E0%B8%A3%E0%B8%9A%E0%B8%B1%E0%B8%8D-big-data-%E0%B8%A0%E0%B8%B2%E0%B8%84%E0%B8%9B%E0%B8%8F%E0%B8%B4%E0%B8%9A%E0%B8%B1%E0%B8%95%E0%B8%B4-fc2c0ccbb696

Apache Spark

https://blog.datath.com/apache-spark-big-data/

Data store

https://fulcrum.rocks/blog/why-use-postgresql-database

https://virtual-dba.com/blog/pros-and-cons-of-mongodb/

https://cloudinfrastructureservices.co.uk/mysql-vs-postgresql/

Apache Airflow

https://blog.skooldio.com/what-is-apache-airflow/

On-Cloud vs On-Premise

https://blog.cloudhm.co.th/on-cloud-vs-on-premise/

https://netway.co.th/kb/blog/cloud-managed-services/cloud-computing-คืออะไร

https://www.nipa.cloud/blog/compare-cloud-in-thailand

https://monsterconnect.co.th/aws-vs-azure-vs-google-cloud/

https://blog.cloudhm.co.th/3-global-cloud/

https://hardcoreceo.co/cloud-computing-intoduction/

https://www.techtalkthai.com/aws-google-microsoft-get-65-percents-in-q1-2022-cloud-computing-market-share/

https://cloudplatform.googleblog.com/2015/04/big-data-cloud-way.html

https://medium.com/@diemasaksyafachriza/data-pipeline-architecture-28687d7dce6b

https://www.whizlabs.com/blog/google-compute-engine-features-and-advantages/

https://www.parallels.com/blogs/ras/app-engine-vs-compute-engine/

https://wizpals.com/composition-strategy/o-level-english-write-the-pros-and-cons

https://www.tangerine.co.th/google-cloud/google-compute-engine/

https://blog.datath.com/google-cloud-platform-1-compute/#Compute_Engine_brikar_hi_chea_khxm_prakxb

https://blog.datath.com/google-cloud-platform-2-storage-database/