Vol.2019-IOT-44 No.16
2019/3/7

Notebook を介した作業ノウハウの継承・移転を分析するための基盤

長久勝[†1]　政谷好伸[†1]　谷沢智史[†2]　中川晋吾[†3]　合田憲人[†1]

概要：筆者らは「Literate Computing for Reproducible Infrastructure」の方法論に基づき、「Jupyter Notebook」を使った、システムの構築と運用を実践している。筆者らの実践の中で Notebook は、日々作成されるもの、使い回されるもの、前提の変遷により使い古されるものが蓄積されていく。これらの Notebook 群において、Notebook 間の再利用や依存といった関係性、Notebook とその操作対象のライフサイクルなどに着目した検索や分析を可能とするワークベンチを試作した。分析基盤を設計するにあたっては、Notebook のメタデータに「meme」と称する UUID を埋め込み、再利用関係を系統として把握できることを目指した。本稿では、構築した「ノートブックライフサイクル分析基盤」について述べ、分析例を示す。

Workbench for analyzing operational know-how propagation and transfer through shared Notebooks

MASARU NAGAKU[†1]　YOSHINOBU MASATANI[†1]
SATOSHI YAZAWA[†2]　SHINGO NAKAGAWA[†3]　KENTO AIDA[†1]

Abstract: “Literate Computing for Reproducible Infrastructure (LC4RI)” is our approach for deploying and managing IT infrastructure. All through our practices Notebooks, which record operational procedures, are constantly created, reused, eventually outdated, and archived. We developed a prototype workbench for understanding Notebook’s life-cycle. It intends to analyze propagations of reuse and dependencies among Notebooks, then to help search reusable candidate in the context of a given target system. For this purpose, “meme” meta-data has been introduced as Jupyter Notebook’s extensions. It assigns unique UUID to every notebook and to each cell within it in order to track dependencies and development histories. This article explains our analytic workbench for Notebook’s life-cycle and its use cases.

1. はじめに

筆者らは「Literate Computing for Reproducible Infrastructure」[1][2]（以下、LC4RI）の方法論に基づき、「Jupyter Notebook」[3]を使った、システムの構築と運用を実践している。具体的には、現在、国立情報学研究所が運用している、OpenStack によるベアメタルクラウドの運用に、LC4RI を用いている。このクラウド基盤は、構築時期の異なる 2 世代の基盤を並行して運用しており、新しい方の基盤[4]では、Notebook ベースの構築手順が整備されている、言わば、LC4RI ネイティブな基盤となっている。

LC4RI による Notebook 利用では、実行セルの出力を含んだ Notebook が作業証跡となるため、その再利用においては、以前の作業証跡を残すために、コピーして使う方針を採っている。この状況下では、Notebook が単調増加するため、再利用したい Notebook を検索する方法が必要となった。また、蓄積された Notebook 群が、対象システムに対する運用作業の実態すべてとなるため、これを統計的に分析することで、対象システムの問題点の洗い出しを行い、運用の見直しや、対象システムの改修につなげられるのではないかと考えた。

こうした問題意識から、Notebook のメタデータに「meme」と称する UUID を埋め込む仕組みを開発し、この上で Notebook を蓄積した後、meme を手掛かりに Notebook 間の関係を分析する基盤を構築した。本稿では、この取り組みについて述べる。

2. LC4RI

LC4RI では、Jupyter Notebook の形式で、情報システムの構築や運用についての手順を、Notebook 群として書き、これを使って、情報システムの構築や運用を行う。また、パッチ適用や未知の障害対応など、運用実務において手順が確立されていない作業の際にも、既存の Notebook を部分的に再利用したり、新規の Notebook を書き起こすなど、Notebook ベースの運用作業を行う。

Notebook で作業を行うことで、全ての作業結果が残されるため、これを作業証跡として残しておく。作業対象を変更するなどの軽微な違いのみで、過去に使った Notebook の再利用を行う場合は、その Notebook のコピーを作成し、このコピーを用いて作業を行う。部分的再利用の場合も同

†1 国立情報学研究所
National Institute of Informatics
†2 株式会社ボイスリサーチ
Voice Research Inc.
†3 有限会社カラビナシステムズ
Carabiner Systems, Inc.

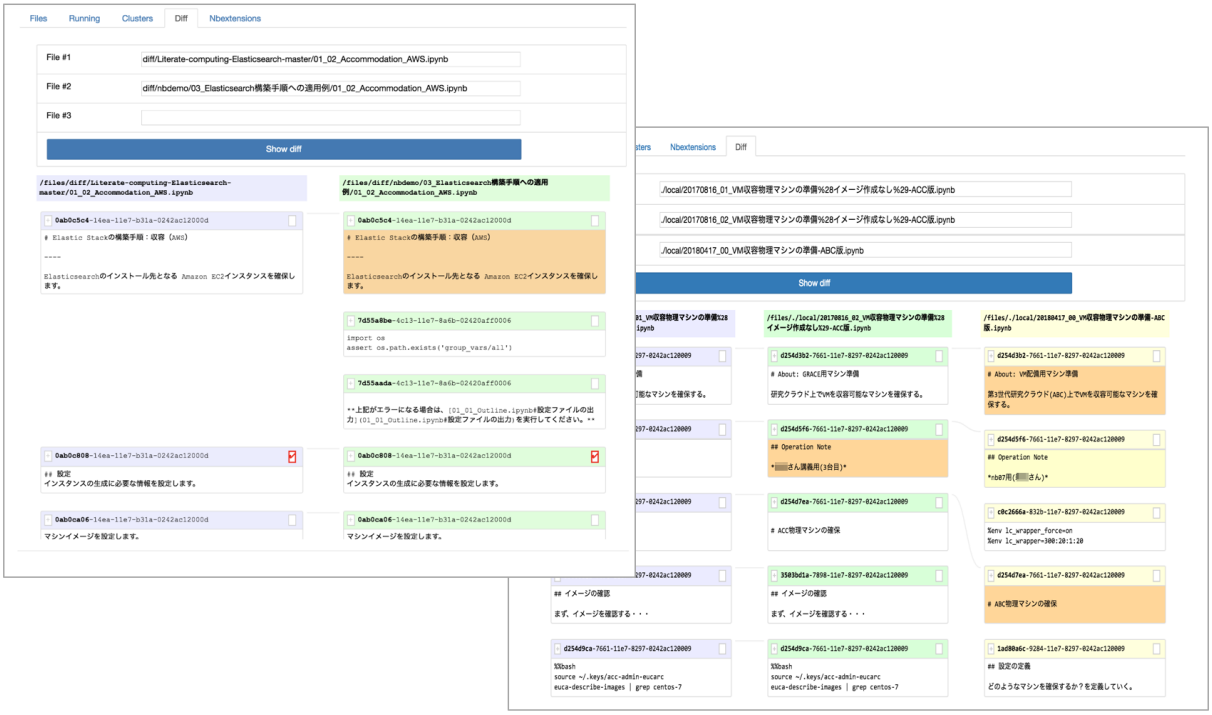

図 1：meme を考慮した diff

様とする。これを続けていくと、同じような手順の作業結果が入った Notebook が、いくつも作成され、蓄積されていく。筆者らは、作業日時と作業内容に基づく Notebook の命名規則を用いており、何度も実施する作業であれば、作業日時の表記だけが異なる Notebook 群が蓄積されていく。

運用対象となる情報システムは、構築された時点から、利用や運用作業によって、少しずつ内部の状態を変えていく。このため、構築時点で正しかった運用手順が、どこかの時点で正しくなくなる可能性がある。例えば、パッケージのアップデートによって、あるコマンドを実行した際のメッセージ出力が変化した場合、メッセージ出力を確認する手順は影響を受ける。このため、Notebook の再利用においては、特定の作業手順のマスターとして書かれた Notebook が存在して、それを毎回使うという考え方ではなく、その作業に使われた過去の Notebook のうち、直近に使われたものをコピーして再利用している。これにより、対象システムが変化していたとしても、もっとも正しく機能するであろう、直近で使われた Notebook の再利用が行える。もし、対象システムの変化によって、その Notebook の一部が正しく機能しない場合は、その場で作業者が調査し、Notebook を修正して、意図した作業を完遂させる。これにより、次回の再利用時に、正しく機能する可能性を高めておくことができる。

対象システムの変化によって、必要な手順が変化した場合に、マスターとして管理された Notebook をメンテナンスするアプローチも可能であるが、著者らの現場では、直近で使われた Notebook の再利用によって、それに替えている。ムービングターゲットである対象システムに対し、コストをかけてマスター手順を管理せずとも、正しく機能する可能性の高い手順を担保できている。

3. Lineage メタデータを埋め込む目的

3.1 Lineage 開発の経緯

meme と読んでいるセル毎にユニーク ID を付与する当初の要件は、我々の wrapper カーネル拡張[5]において操作ログを取得する際に当該の出力に対応するセルを特定したかったことに端を発する。

その後、Notebook を改変した際の差分を Notebook の構造を考慮して合目的的に表示したい[6]（図 1）、さらには Notebook 間の再利用や依存といった関係性、Notebook とその操作対象のライフサイクルを理解するための手がかりとなるメタデータを取得できるようにしたいと考えるようになった。

その結果、ユニーク ID としての meme について、Notebook 上の空間的な、Notebook の再利用による時間的な、関係性を履歴として保持できるような「Lineage メタデータ」を実装するに至った[7]。

3.2 Notebook のライフサイクルを理解する

Notebook のライフサイクルとして、筆者らの実践の中では以下のようなユースケースを想定している。

- Notebook は作業単位毎に作成される。
- 作業単位の区切りとして、内容によるものだけでなく、個人による作業が途中で日を跨ぐなど中断される場合、作業を別の作業者に途中で引き継ぐ場合なども、区切りとする。区切り毎に Notebook を保存し、再開時にはコピーを新たに作成して、作業を続けるものとする。中断した Notebook をそのまま続けて使うことはしない。
- 過去の作業と類似の作業を行う場合には、過去の Notebook のコピーを新たに作成し、そのコピーを改変・再利用しながら作業する。過去の Notebook には作業証跡が含まれているため、直接の再利用はしない。
- Notebook は、類似・共通の作業であっても作業対象毎に分割して作成する。例えば、正副のコントローラに対しパッチを当てる場合、1 つの Notebook にまとめて記述するのではなく、正用の作業 Notebook と副用の作業 Notebook に分けて作成する。こうした状況の多くの場合、1 つ目の Notebook で作業を終えた後、その Notebook をコピーして 2 つ目以降の作業を行う。
- 現行の実績のある作業、それを実施した Notebook を正本と見なす。つまり特定の作業対象に対して、試行錯誤ないし作業改善が行われた場合、直近で作業完了（作業の目的が達成された）した際の Notebook を正本と見なす。
- 作業内容に幅がある場合、その幅をカバーする万能の Notebook を書くのではなく、派生を許容する。作業対象（例えばサーバ）の振る舞いが、時間を経て均一でなくなり、かつ、その変化を許容すると判断した場合、作業対象と対となる派生 Notebook が生まれることを許容し、元手順の Notebook と派生 Notebook のいずれも正本と見なす。

筆者らの実践の中で Notebook は、生まれ、再利用され、場合によっては使われなくなる。こうして蓄積されている Notebook 群に対して、Notebook 間の再利用や依存といった関係性、Notebook とその操作対象のライフサイクルが存在していると考えられる。これに着目した検索で、再利用したい Notebook を見つけたり、作業者の Notebook の使い方に着目して分析することで、チームの状況を確認することが可能であろう。こうした検索、分析のシステム化を実現するために Lineage メタデータを設計した。

4. Notebook における Lineage メタデータ実装

一般的に拡張子を「.ipynb」とする Jupyter Notebook 形式の Notebook は、JSON 形式で書かれたテキストファイルである。プラグインなどの拡張により、Notebook に情報を追加したい場合、メタデータとして付与することができるように設計されており、文書に対する付与とセル単位での付

```
...
"lc_cell_meme": {
  "current": "8f5c5fe2-71cc-11e7-9abe-02420aff0008",  ## The meme id for this cell
  "previous": "8f5c5cea-71cc-11e7-9abe-02420aff0008", ## The context of this cell as reference to the previous cell
  "next": "f2125b84-4669-11e7-958b-02420aff0006",      ## The context of this cell as reference to the next cell
  "history": [                                        ## The history of this notebook's context
    {
      "current": "8f5c5fe2-71cc-11e7-9abe-02420aff0008",
      "previous": "8f5c5ee8-71cc-11e7-9abe-02420aff0008",
      "next": "8f5c60aa-71cc-11e7-9abe-02420aff0008"
    },
    {
      "current": "8f5c5fe2-71cc-11e7-9abe-02420aff0008",
      "previous": "f2125b84-4669-11e7-958b-02420aff0006",
      "next": "8f5c5ee8-71cc-11e7-9abe-02420aff0008"
    }
  ]
},
...
```

図 2：lc_cell_meme の例

```
...
"lc_notebook_meme": {
   "current": "1904d564-71c6-11e7-8369-0242ac110002", ## The meme id for this notebook
   "lc_server_signature": {
      "current": {
         "server_url": "https://xxxxx.nii.ac.jp/user/xxxx/", ## The current notebook server infomation is also stored
         "notebook_path": "/",
         "notebook_dir": "/notebooks",
         "signature_id": "034b406c-71c8-11e7-a8bf-02420aff0008"
      },
      "history": [ ## Travel history where this notebook has been executed
         {
            "notebook_dir": "/notebooks",
            "notebook_path": "/",
            "server_url": "http://localhost:8888/",
            "signature_id": "dc3b0162-71bb-11e7-8369-0242ac110002"
         }
      ]
   },
},
...
```

図 3：lc_notebook_meme の例

与が可能である。Lineage メタデータもメタデータとして記録されるように実装されており、Notebook に対する Lineage メタデータを文書に対するメタデータ「lc_notebook_meme」として、セルに対する Lineage メタデータをセルに対するメタデータ「lc_cell_meme」として、それぞれ記録される。Lineage メタデータが記録されるタイミングは、Notebook が保存（自動保存含む）されるタイミングであり、meme がまだない場合の付与や、後に述べる状況変化に応じた情報の更新が行われる。

図 2 に示すように、セルに対する lc_cell_meme は、「current」「previous」「next」の 3 つの UUID と、この 3 つ組の配列「history」で構成される。current は、そのセル自身を識別するための UUID である。Notebook 上のセルは、シーケンシャルに配置されているので、セルの間には前後関係が存在する。この関係を記録するために、次のセルの UUID を next に、前のセルの UUID を previous に記録している。Notebook 内で、先頭のセルの previous、最後尾のセルの next には、「null」が記録される。Jupyter Notebook 上でセルのコピー操作を行った場合、メタデータもコピーされるため、current がコピーされ、同じ meme を持つセルが作成される。previous、next については、Notebook の保存時に、前後の状態に合わせて更新される。Notebook の保存時に、保存前に保持していた 3 つ組と、現状が異なる場合は、その更新に加えて、以前の 3 つ組を history に追加し、セル自身が前後関係についての履歴を保持する設計となっている。現状の lc_cell_meme の実装では、meme の払い出しとセルの内容に紐付きがないため、コピーされたセルにまったく異なる記述を上書きすると、意味的なつながりのないセルが同じ meme を持ってしまう問題がある。しかし、作業対象の IP アドレスを変数に設定しているセルなど、意味は変わらなくても、機械的に記述内容の同一性を見る（例えばハッシュ値など）ことが困難な場合が多いため、Notebook の書き方として注意するに止まっている。

図 3 に示すように、Notebook に対する lc_notebook_meme は、その文書自身を識別する UUID「current」と、Notebook が使われた環境に関する情報を保持する「lc_server_signature」内に、現在の状況「current」と、履歴を保持する「history」で構成される。lc_notebook_meme の current は、Notebook のコピー（ファイルのコピー）で引き継がれる。lc_server_signature には、Notebook が使われた環境における、Notebook の置かれた場所「notebook_dir」「notebook_path」、Jupyter Notebook サーバの URL「server_url」、Jupyter Notebook サーバが動作している環境を識別する ID「signature_id」、が保持され、これに変化があった場合には、以前の情報を history に追加し、lc_server_signature を更新する。筆者らの実務環境においては、Jupyter Hub ベースで、ユーザ毎に個別の作業環境を与える「Operation Hub」と名付けた Juputer Notebook 環境を

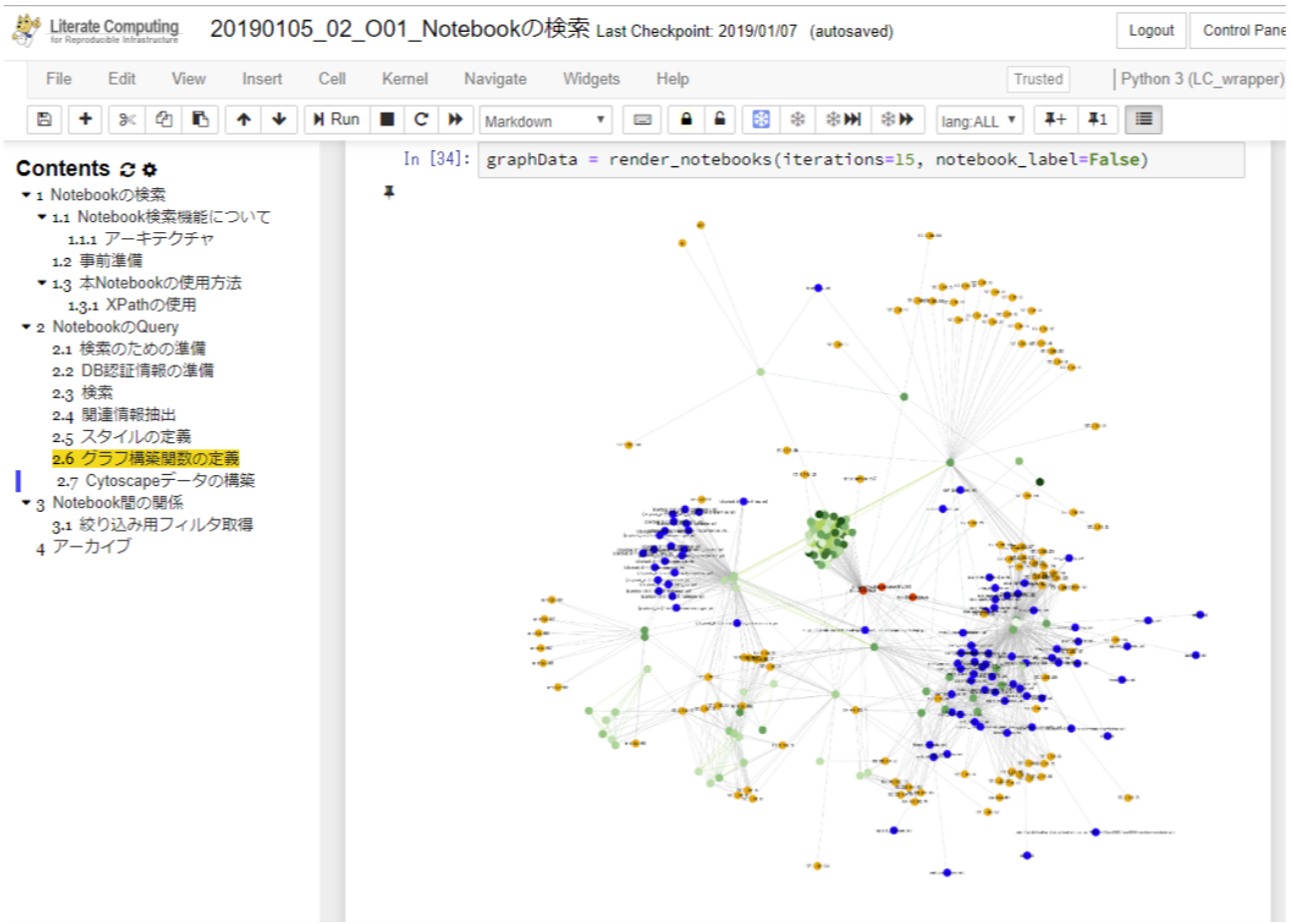

図 4：分析基盤の実行例

構築し利用しており、登録された作業者に対し一意となる signature_id を紐付けることで、どのユーザの環境で Notebook が使われたのか、追跡できるようにしている。

なお、Notebook 内の Lineage メタデータを消去して一新したい場合などに備えて、Notebook 内の meme を操作するためのコマンドラインツールも開発しており、構築手順を記した Notebook 群一式を外部に提供する場合などに使用している。

5. meme に基づく「ノートブックライフサイクル分析基盤」

meme を手掛かりに Notebook 群を分析しやすい状況が作れたとした上で、実際に Notebook 群を分析する方法を考える。この時、目標として、Notebook 群に対して特定の指標が得たいというだけではなく、クラスタなどの構造化データを含め自由に分析結果を得たい、特定の条件に当てはまる Notebook を検索したい、といったユースケースも満たしたいと考えた。この多様性を担保するためには、特定の分析処理の実装を行うだけでは不十分であり、さまざまな問い合わせに対応可能な、Notebook 群をデータベース化した分析基盤を構築し、その上で分析処理を実装できるようにする必要があると考えた。

Notebook 群をデータベース化するにあたっては、Notebook が、JSON 形式で、再帰的な key-value 構造を持っていることに着目した。こうした構造を扱えるデータベースとして、Mongo DB などを挙げることができる。しかし、いわゆる KVS では、スキーマレスなデータを扱いやすい反面、問い合わせの表現能力が弱いことが懸念された。そこで、XPath による強力で標準化された問い合わせの採用を検討し、Notebook を XML 化して PostgreSQL に投入することとした。

PostgreSQL に XPath で問い合わせを行い、計算処理を行う部分については、Jupyter Notebook を採用した。

図 4 に分析基盤の実行例を示す。ここでは、Hadoop 関連システムの構築と運用に使われた Notebook 群に対し、誰がどの Notebook で作業したか、それぞれの Notebook の作業で使われた資材（Ansible の Playbook など）はどれか、それぞれの Notebook の作業対象となったマシンはどれか、といった情報を、XPath による問い合わせ結果を Python で処理し、図示している。

緑色のグラデーションで表された Notebook の塊は、保有する meme の類似度が高いものの一群を表しており、コ

ピーによって再利用された、ほぼ同じ内容の Notebook と考えられる。同じ群に含まれる別の Notebook に対して、異なる作業者（赤丸）が紐付いており、Notebook を介して別の作業者が同じ内容の作業を行ったことが見てとれる。また、それぞれの Notebook に紐付いたサーバ（黄丸）の群が異なることから、異なる Hadoop クラスタに対する作業が行われたことも分かる。

6. おわりに

本稿では、LC4RI の実践で蓄積された Notebook を検索・分析する取り組みについて報告した。現在も、日々の運用で Notebook は蓄積され続けており、今後もさまざまな分析が可能であると考えている。

今回の分析結果は例示に過ぎないが、Notebook を介して別の作業者が同じ内容の作業を行ったことが確認でき、Notebook を介した作業ノウハウの継承・移転が起こっている可能性を示していると考えられる。こうした分析は、LC4RI の効果測定につながると考えている。

また、図の上に類似度で配置された Notebook について、作成・利用された日付で並べると、再利用時のコピーの関係を樹形図状に示せる。コピー関係の先端部の集合を、正本 Notebook 群と見なすことで、検索や、Notebook のカタログ化など、作業者支援につなげることも考えられる。

今後も、この分析基盤を活用し、人間中心の、より安定した、情報システムの構築・運用につなげたい。

参考文献

[1] Masatani, Y.. Collaboration and automated operation as literate computing for reproducible infrastructure. JupyterCon, 2017-08-25.
[2] 長久勝, 政谷好伸, 谷沢智史, 中川晋吾, 合田憲人. Literate Computing for Reproducible Infrastructure による研究・教育環境の構築と運用. 大学 ICT 推進協議会 2017 年度年次大会, 2017-12-13.
[3] “Project Jupyter”. https://jupyter.org/, (参照 2019-02-04).
[4] 長久勝, 政谷好伸, 横山重俊, 谷沢智史, 中川晋吾, 合田憲人. Academic Baremetal Cloud の実現. 大学 ICT 推進協議会 2018 年度年次大会, 2018-11-19.
[5] “Jupyter-LC_wrapper”. https://github.com/NII-cloud-operation/Jupyter-LC_wrapper, (参照 2019-02-04).
[6] “Jupyter-LC_notebook_diff”. https://github.com/NII-cloud-operation/Jupyter-LC_notebook_diff, (参照 2019-02-04).
[7] “Jupyter-LC_nblineage”. https://github.com/NII-cloud-operation/Jupyter-LC_nblineage, (参照 2019-02-04).