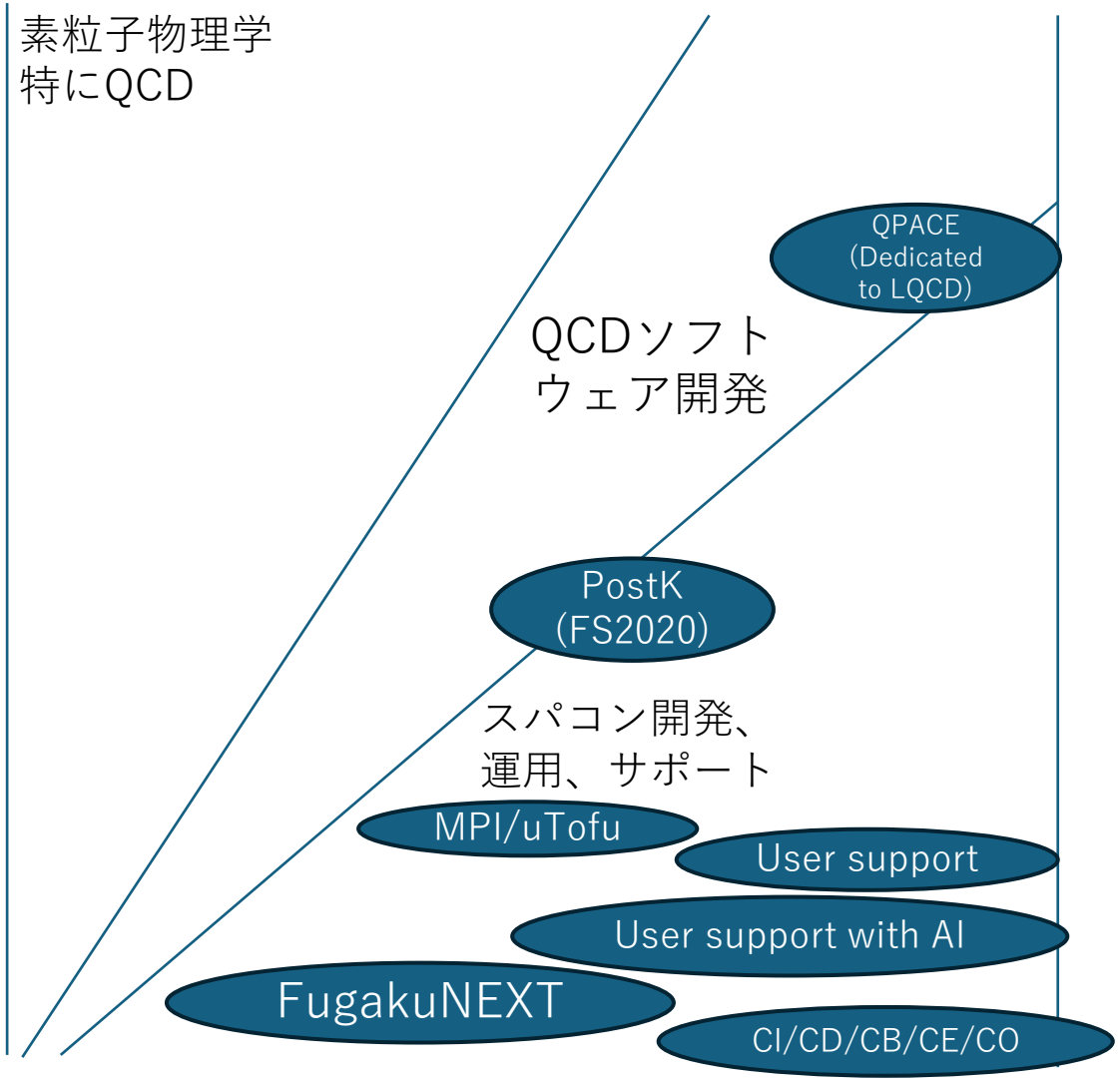
CXフレームワークによるアプリケーション開発

中村宜文
理化学研究所計算科学研究センター
運用技術部門ソフトウェア開発技術ユニット
富岳NEXT 通信ライブラリWGリーダー・ベンチマークWGサブリーダー

2025年9月3日
第25回 High Performance Computing Physics (HPC-Phys) 勉強会

自己紹介



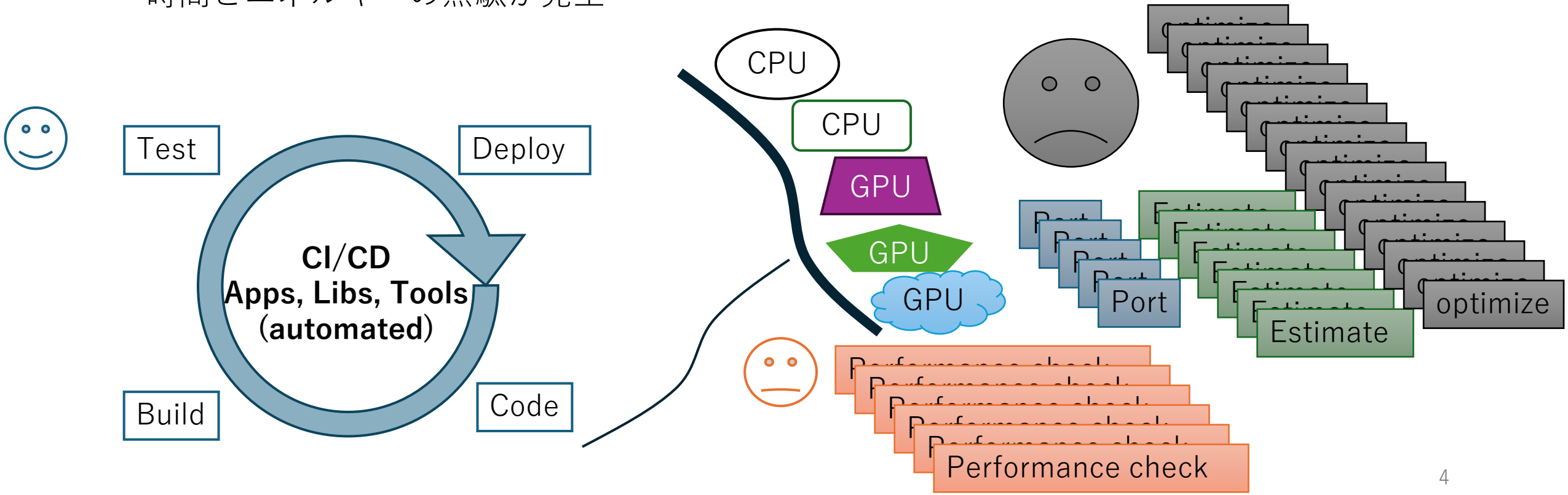
- ドイツ電子シンクロトロン（DESY）
  - 2005 – 2008
- レーゲンスブルク大学
  - 2008 – 2010
- 筑波大学
  - 2010 - 2011
- 理化学研究所
  - 2011 –

CXフレームワークとは

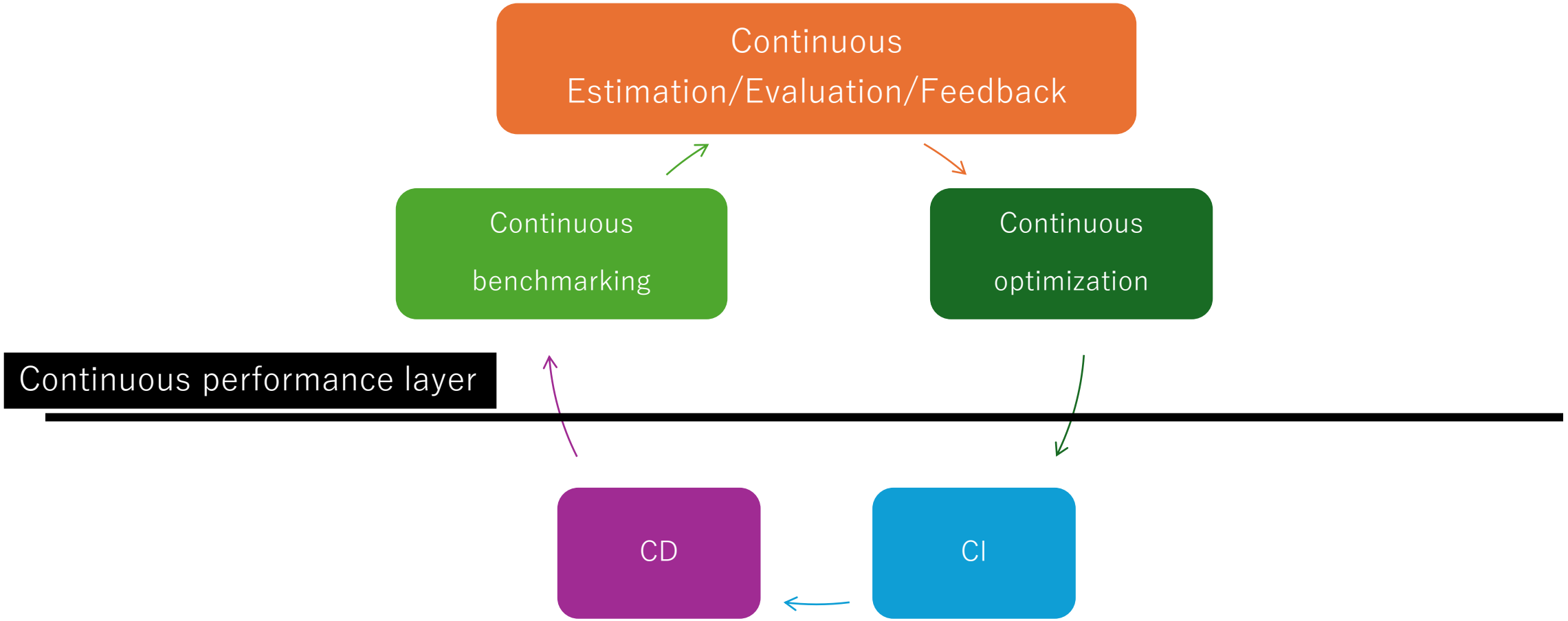
- ソフトウェアの継続的インテグレーション/デリバリー/デプロイ(CI/CDフレームワーク)をソフトウェア性能を重視して拡張
  - 継続的ベンチマーク (CB)
  - 継続的評価/性能推定/フィードバック (CE/CF)
  - 継続的最適化/強化 (CO/CE)

なぜCI/CDだけではHPCには不十分か

- 今日、ソフトウェア開発においてCI/CDは一般的になっている。しかし、、
- 性能チェックが後回しで手動になりがち
- 時間とエネルギーの無駄が発生



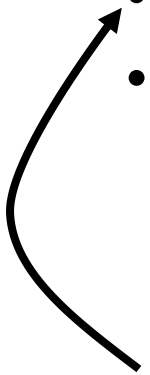
継続的に性能を高める開発



ゴール：性能確認と改善を通常の開発フローに組み込む

継続的に性能を高める開発の原則

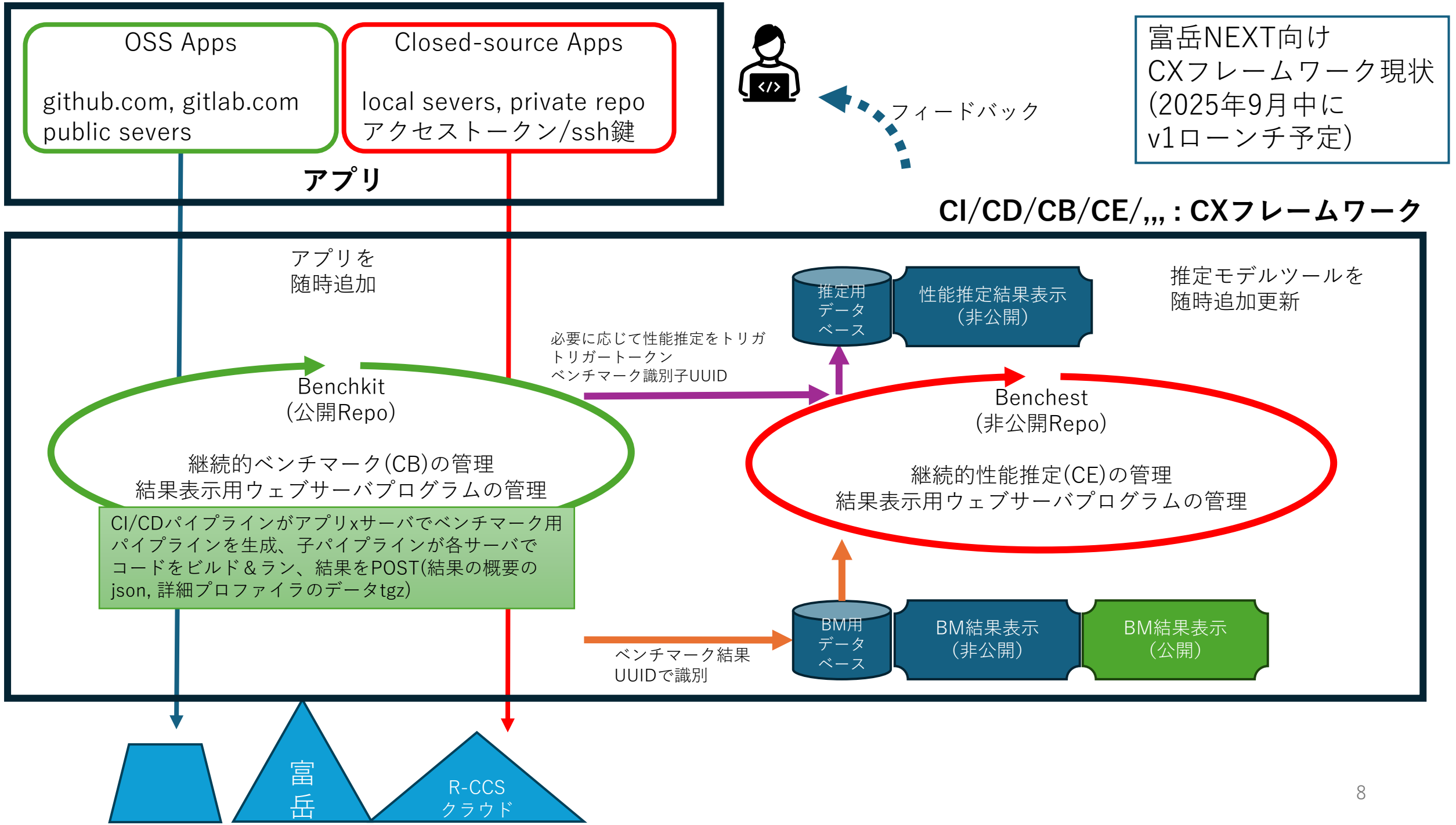
- The 2nd “FugakuNEXT” Application Seminar /第２回「富岳NEXT」アプリケーションセミナー, July 17, 2025で４つの原則を示しました。その時のスライドを本講演のスライドの後半に追加しておきます。
  - Principle 1: Meaningful & Validated Performance Measurement
  - Principle 2: Continuous Performance Monitoring & Feedback
  - Principle 3: Continuous Optimization
  - Principle 4: Open, Flexible, and Sustainable Integration



AIによる自動最適化についてはいくつか報告がなされています。

富岳NEXT向けCXフレームワーク

- 多くのアプリケーションが参加できる
  - 非公開ソフトウェアも参加できる仕組み
    - アクセストークン
    - ssh公開鍵
    - ソースコードの受け渡しは無しで、ビルド済み実行ファイル&インプットだけ
    - など、、
- ベンチマーク性能を公開
  - 非公開にも応じられる
- 性能推定
  - 富岳NEXT向け性能推定ツール（理研、富士通、NVIDIA）
- ロードマップ
  - 第１版：2025年9月中
  - （第２版：性能推定機能の追加/強化、年内？？）
  - （第３版：生成AIによる自動最適化、年度内？？）


OSS Apps
github.com, gitlab.com
public severs
Closed-source Apps
local severs, private repo
アクセストークン/ssh鍵
アプリ
フィードバック
富岳NEXT向け
CXフレームワーク現状
(2025年9月中に
v1ローンチ予定)
CI/CD/CB/CE/,,, : CXフレームワーク
アプリを
随時追加
推定用
データ
ベース
性能推定結果表示
(非公開)
推定モデルツールを
随時追加更新
必要に応じて性能推定をトリガ
トリガートークン
ベンチマーク識別子UUID
Benchkit
(公開Repo)
継続的ベンチマーク(CB)の管理
結果表示用ウェブサーバプログラムの管理
CI/CDパイプラインがアプリxサーバでベンチマーク用
パイプラインを生成、子パイプラインが各サーバで
コードをビルド＆ラン、結果をPOST(結果の概要の
json, 詳細プロファイラのデータtgz)
Benchest
(非公開Repo)
継続的性能推定(CE)の管理
結果表示用ウェブサーバプログラムの管理
BM用
データ
ベース
BM結果表示
(非公開)
BM結果表示
(公開)
ベンチマーク結果
UUIDで識別
富岳
R-CCS
クラウド

benchkit

https://github.com/RIKEN-RCCS/benchkit
(旧:https://gitlab.swc.r-ccs.riken.jp/fugakunext/benchmark/benchkit)
富岳NEXT ベンチマークWGで開発しています。

- 機能
  - 多拠点でのベンチマーク（**Jacamar-CI**を利用してjobを管理します。）
  - 結果のアップロード、結果の表示、性能推定(benchest)のトリガ
  - **Spack**利用（ベンチマーク実行拠点で準備されているものを利用する想定）
    - パッケージ管理ツール
  - **Benchpark**利用（ベンチマーク実行拠点で準備されているものを利用する想定）
    - 内部でSpack, Ramble（ジョブの管理など）を利用
- ベンチマーク拠点（現：富岳、R-CCS Cloud）
  - 拠点追加
    - 方法1. benchkit関係者にジョブが投げられるアカウントを付与
    - 方法2. Singularityコンテナを利用して、環境構築
      - gitlab-runner token受け渡しなどが必要。手順整備中

富岳、R-CCS Cloudでのベンチマークは、
私のアカウント・バジェットで動きます

- ベンチマークアプリケーション
  - アプリケーション追加
    - 手順（https://github.com/RIKEN-RCCS/benchkit/blob/main/ADD_APP.md）
    - オープンソースソフトウェアでなくても対応可能です。非OSS向け手順準備中
  - **性能指標（FOM: Figure of Merit）の定義（と出力）が必要です。**
    - アプリの出力からFOMを抜き出して、指定のファイルに指定の形式で出力する。
    - 確度の高い性能推定のため、特性が異なる区間ごと（演算、通信）の測定（と出力）が必要です。

Uploaded Results

Search by multiple keyword

Timestamp	CODE	Exp	FOM	FOM version	SYSTEM	Nodes	JSON	PA Data
2025-09-03 10:11:08	qws	DummyFrom_qc-gh200	11.22	dummy	RC_GH200	1	json	-
2025-09-03 10:11:03	genesis-nonbonded-kernels	Oct_mod1	14.781	null	FugakuLN	1	json	-
2025-09-03 10:11:03	genesis-nonbonded-kernels	Generic	28.057	null	FugakuLN	1	json	-
2025-09-03 10:11:03	genesis-nonbonded-kernels	Oct	26.588	null	FugakuLN	1	json	-
2025-09-03 10:10:59	genesis-nonbonded-kernels	Generic	15.102	null	Fugaku	1	json	-
2025-09-03 10:10:59	genesis-non						json	-
2025-09-03 10:10:59	genesis-non						json	-
2025-09-03 10:10:54	FS_qws						json	data
2025-09-02 15:40:56	genesis						json	-
2025-09-02 15:36:46	genesis	null	549.248	null	FugakuLN	1	json	-
2025-09-02 15:35:52	qws	CASE7	0.567	DDSolverJacobi	FugakuCN	2	json	-
2025-09-02 15:35:48	qws	CASE0	0.385	DDSolverJacobi	FugakuCN	1	json	-
2025-09-02 15:35:48	qws	CASE1	0.451	DDSolverJacobi	FugakuCN	1	json	-
2025-09-02 15:35:47	qws	CASE0	48.811	DDSolverJacobi	FugakuLN	1	json	data

現在は非公開サーバに置いています。
本番環境では、
　公開サーバに置きます。
　高機密アプリ向けにデータを公開しないオプションを入れます。

富岳NEXT向け
CXフレームワーク将来

OSS Apps

github.com, gitlab.com
public severs

Closed-source Apps

local severs, private repo
アクセストークン/ssh鍵

アプリ

フィードバック？

CI/CD/CB/CE/,,, : CXフレームワーク

アプリを
随時追加
↓
自動追加？

継続的に
フィードバック
する仕組み

継続的に
推定モデルを改良
する仕組み

継続的に
ベンチマーク
を取る仕組み

継続的に
性能推定する
仕組み

継続的に
生成AIで最適化
する仕組み

推定用
データ
ベース

性能推定結果表示
（非公開）

BM用
データ
ベース

BM結果表示
（非公開）

BM結果表示
（公開）

データは人類の資産。公開を目指したい

富岳

R-CCS
クラウド

まとめ

- CXフレームワークによるアプリケーション開発について
- 富岳NEXT向けCXフレームワークの現状
- 富岳NEXT向けCXフレームワークの将来

アプリケーションの性能に軸を置いたCXフレームワークを育てていくために、
アプリ開発者のみなさまからのお声が何より大切です。
ぜひ率直なご意見をお聞かせください。

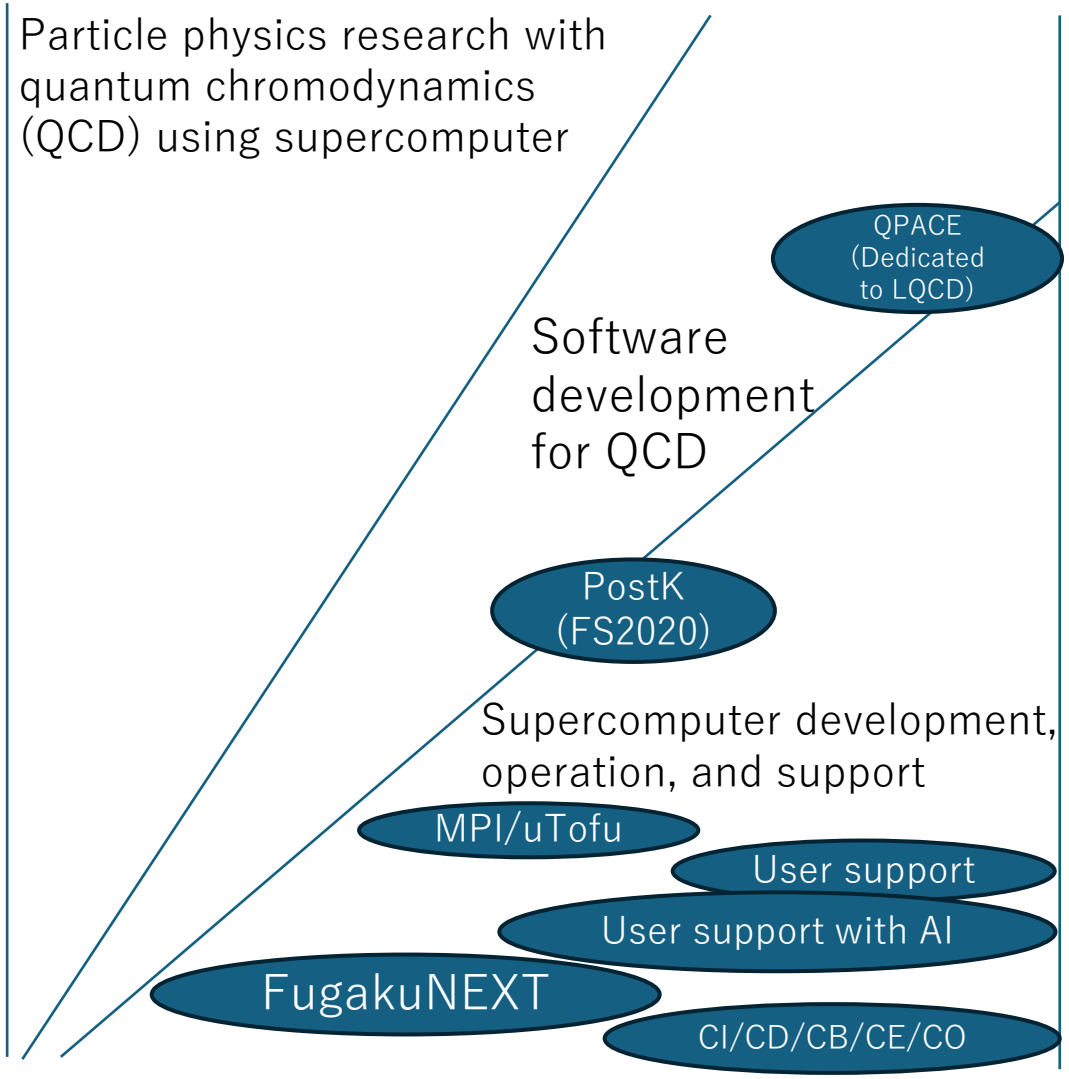
ご清聴ありがとうございました。

Beyond CI/CD: Continuous Performance-Driven Principles for Future HPC Systems and Applications

Yoshifumi Nakamura
RIKEN R-CCS
Operations and Computer Technologies Division, Software Development Technology Unit
FugakuNEXT project, Communication library WG Leader / Benchmark WG Sub-leader

The 2nd “FugakuNEXT” Application Seminar /第２回「富岳NEXT」アプリケーションセミナー
July 17, 2025

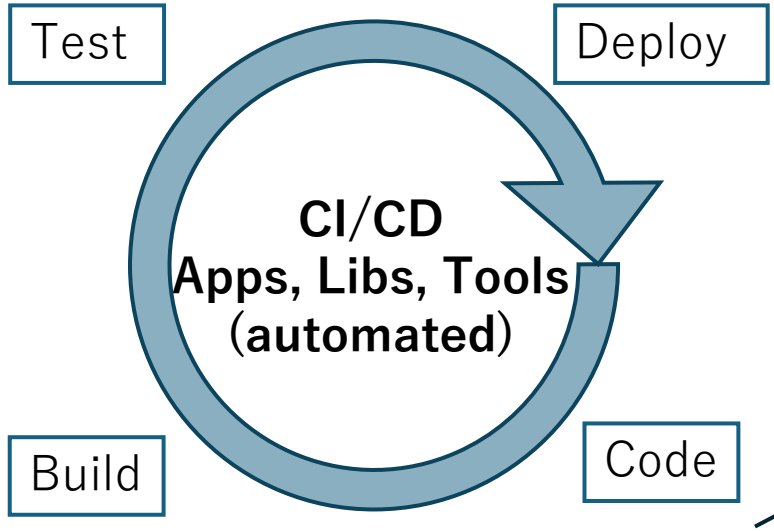
About my work history

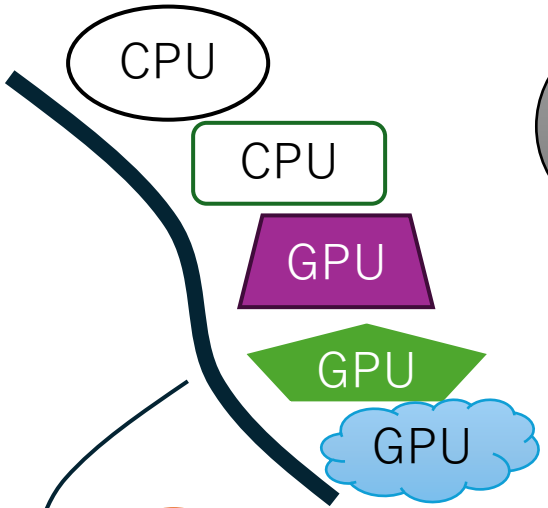


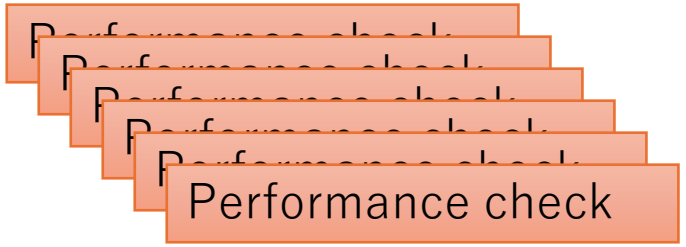
- Deutsches Elektronen-Synchrotron DESY
  - 2005 – 2008
- University of Regensburg
  - 2008 – 2010
- University of Tsukuba
  - 2010 - 2011
- RIKEN
  - 2011 –

Why CI/CD Alone Is Not Enough for HPC

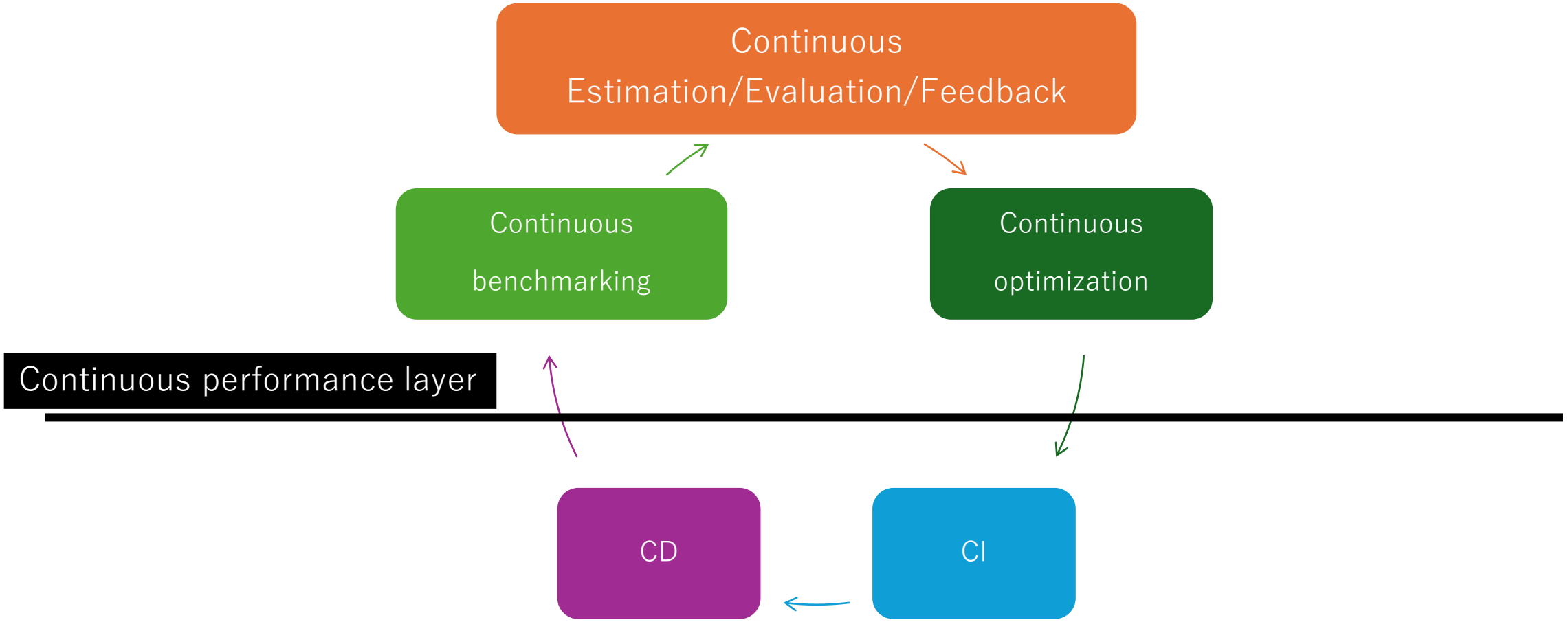
- Continuous Integration/ Continuous Delivery/Deployment (CI/CD) is common
- Performance checks are late, manual
- Wastes time & energy







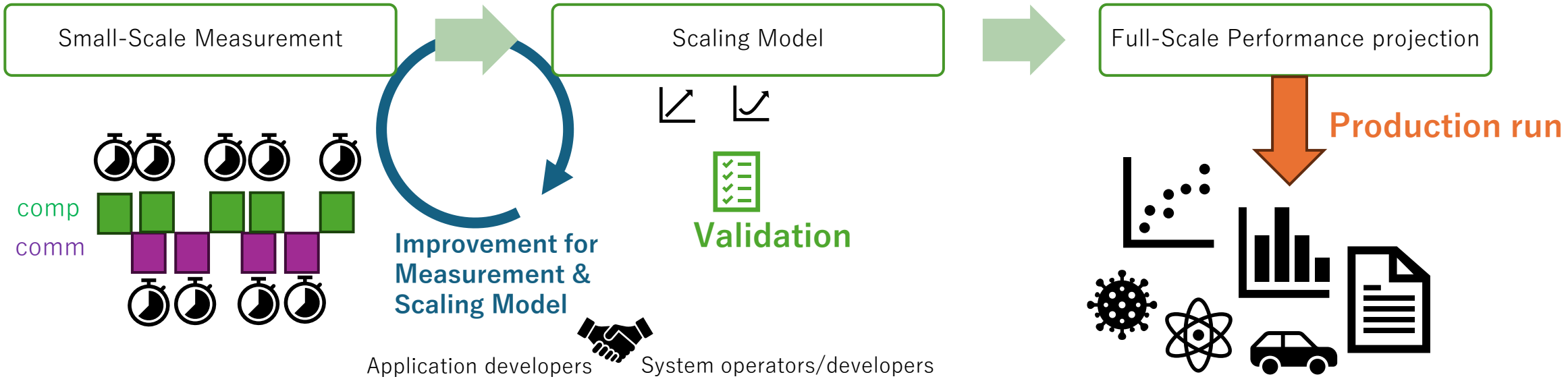
Continuous Performance-Driven Principles



Goal: Performance awareness and improvement become routine, not special isolated tasks.

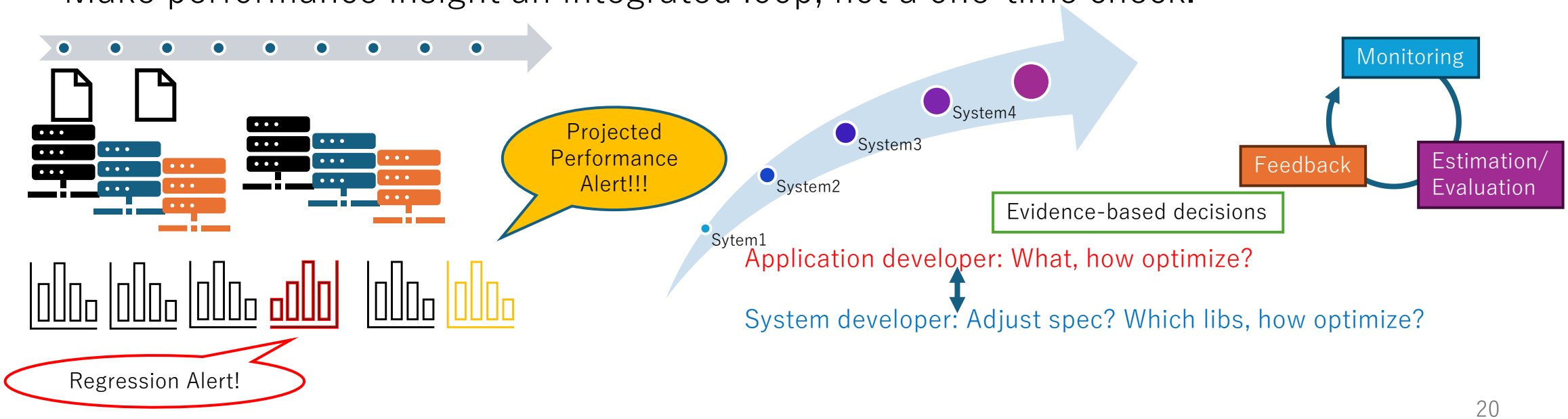
Principle 1: Meaningful & Validated Performance Measurement

- Measure performance in sections matching optimization scopes.
- Use realistic or properly scaled input conditions.
- Ensure performance measurement results are credible for real production.
- Support trustworthy projections for future HPC systems.



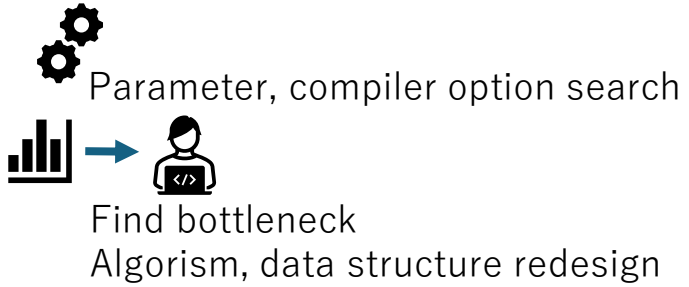
Principle 2: Continuous Performance Monitoring & Feedback

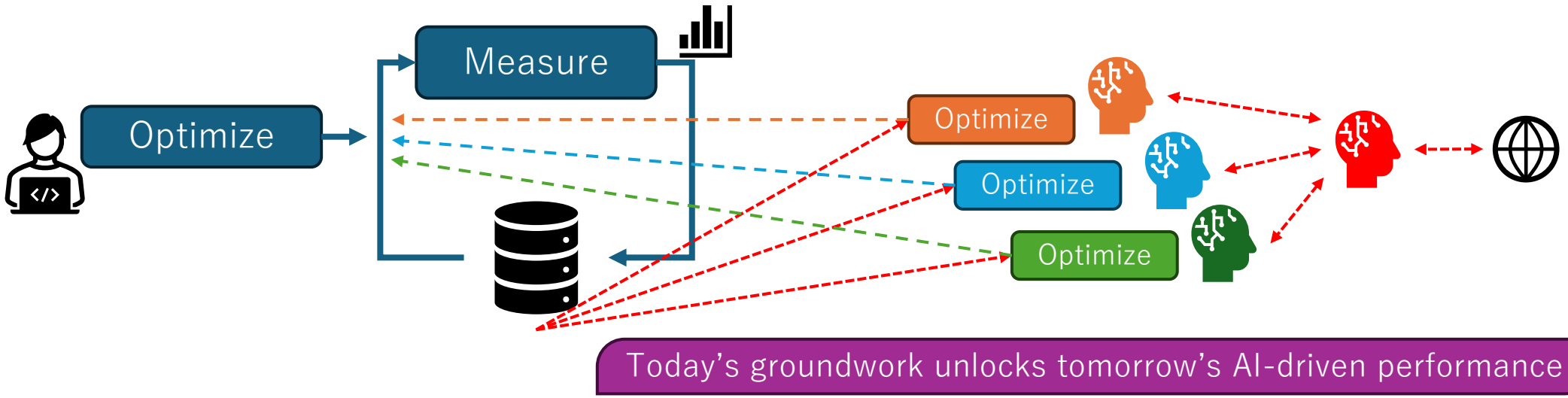
- Continuously collect and observe performance data during real workloads.
- Evaluate results to detect deviations and unexpected trends.
- Estimate future performance under realistic conditions.
- Provide clear, actionable feedback to both application and system teams.
- Make performance insight an integrated loop, not a one-time check.



Principle 3: Continuous Optimization

- Optimization must shift from one-time to continuous.
- Start today: measure more, test variations, record results.
- Automate what you can, guide developers with clear data.
- Build a performance history for AI auto-tuning.



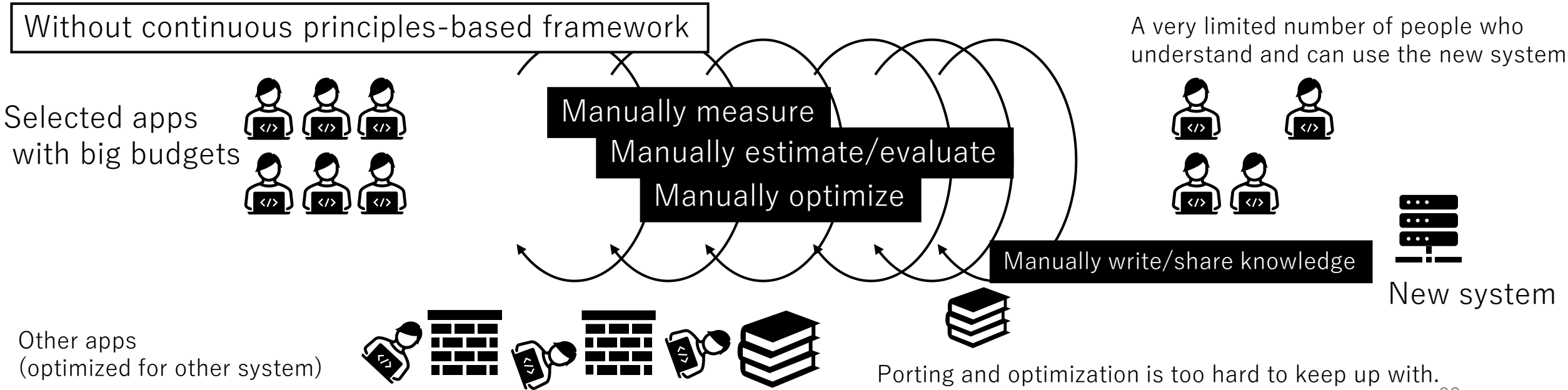


Principle 4: Open, Flexible, and Sustainable Itegration

- Support diversity: HPC systems, tools, teams
- Design for change: Stay adaptable, avoid lock-in
- Enable reuse: Open formats & clear interfaces

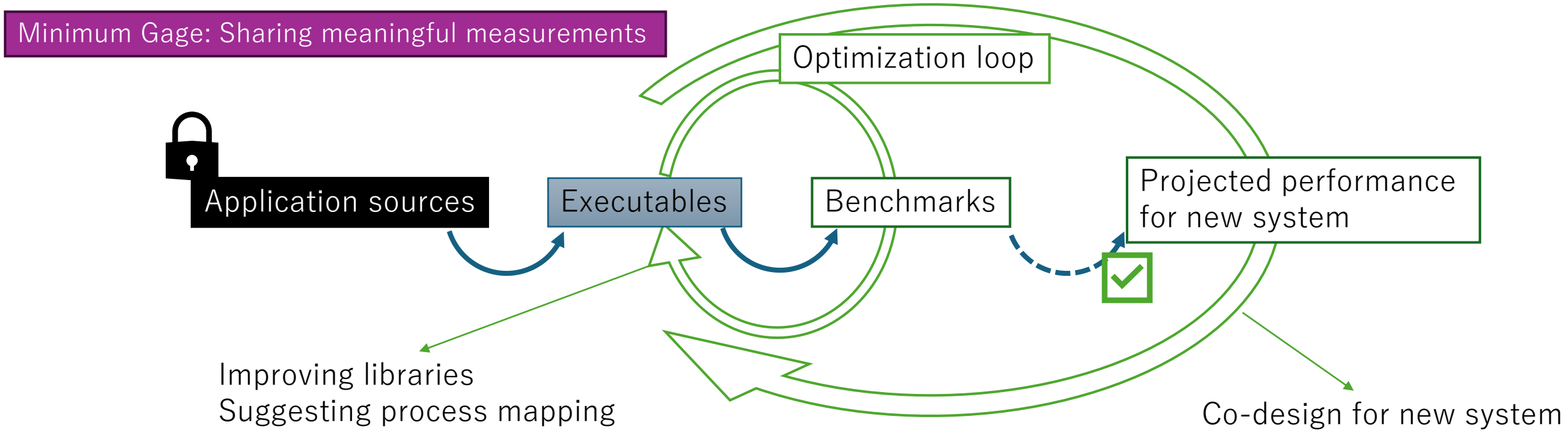
Why This Matters Now for Supercomputer Development

- Today's supercomputers are more complex and use more mixed hardware than ever.
- Hardware and software are still often designed separately. It causes problems.
- We need performance checks and feedback from the beginning.
- This helps design hardware that really fits real apps and workloads.



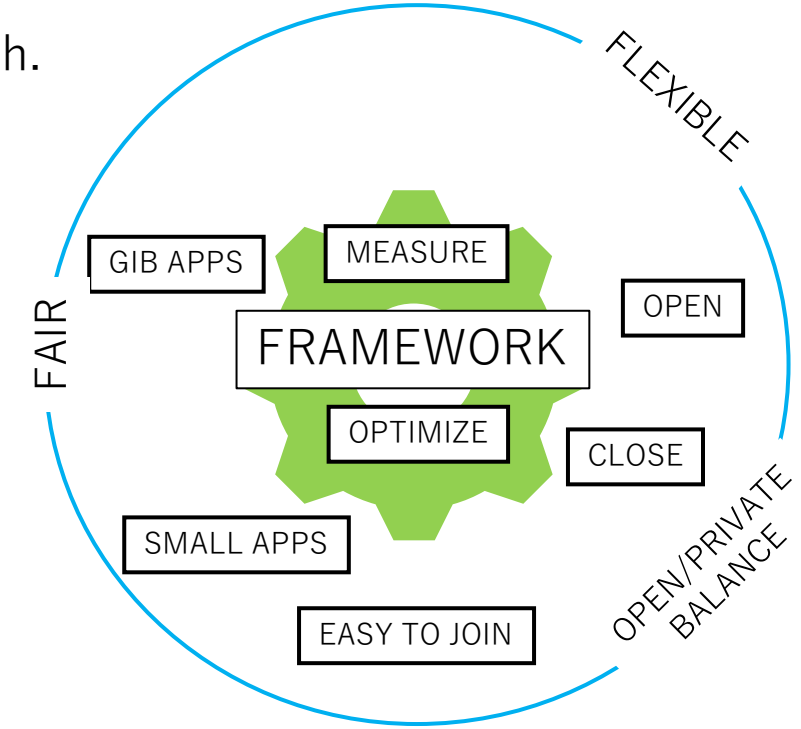
Collaborate and Protect: Open Performance, Private Code

- Full source openness isn't always possible, but sharing key performance data is.
- By sharing benchmarks, both open and closed projects stay in the optimization loop.



Making Performance Work for Everyone

- Many developers are not performance experts.
- The framework should measure and share useful data easily.
- Sharing all code is not needed, sharing performance just enough.
- It should work with little extra effort or skills.
- Openness and privacy must balance.
- Make co-design simple and fair for big and small projects.
- Joining late should still help and benefit all.



Conclusion

- CI/CD alone is not enough, go beyond.
- Add benchmarking, estimation, evaluation, feedback, and optimization.
- 4 principles guide sustainable progress.
- Big or small, open or closed, all apps can grow with HPC.
- We should review these principles as technology and needs change.

Thank you for your attention