WHITEPAPER

賢いアプリが、業務のAI活用を変える—AI PCで実現するローカル実行戦略の核心

Lunar Lake搭載PCによるSLM検証と、Model Hubを核としたアプリ拡張構想

Table of Contents

Executive Summary

生成AIは、すでにクラウド経由での業務利用が企業現場に浸透し始めています。しかし近年、その運用においては継続的なAPIコスト、応答速度の体感的な遅延、機密保持上の制約といった新たな課題が明確になりつつあります。こうした背景のもと、PC端末側にAIモデルを実装し、クラウドに依存せずにローカルでAI推論を完結させる「AI PC」は、柔軟で持続可能なAI活用の基盤として注目を集めています。

一方で、「AI PC」という言葉が流通し始めてからすでに1年以上が経過する中で、実際の業務活用に向けた具体的な設計方針や評価指標は、いまだ明確に定まっていないのが現状です。多くの企業や開発者にとって、「何ができるのか」「どう使えば業務にフィットするのか」「ローカル実行は実用に足るのか」といった問いに対する答えは、十分に可視化されていない状況にあります。

本ホワイトペーパーでは、インテル® Core™ Ultra 7 プロセッサー（開発コードネーム：Lunar Lake）を搭載したHP社製EliteBook X G1iを用い、ローカル環境におけるAIモデル実行の性能・品質・電力効率を多角的に評価しました。対象としたのは、Llama3 Elyza（8B）、DeepSeek R1（14B）、Starcoder2（15B）といった現行世代のSLM（Small Language Model）であり、すべてINT4（4bit整数型）へ量子化されたモデルを、内蔵GPUおよびNPU上で動作させています。

検証は、自然言語処理・コード生成・セキュリティレビューなど20のビジネスタスクを通じて実施され、出力品質、生成速度、First Token Latency、平均電力、W/token（1トークン生成あたりの電力効率）といった指標を用いて、GPUとNPUの比較評価を行いました。その結果、L3 ElyzaをGPU上で実行する構成が、品質・速度・消費電力すべての面で最も高い実用バランスを示すことが明らかとなりました。一方、NPUは短文要約や感情分析など、生成量の少ないタスクにおいては平均消費電力6W未満という高い省電力性を発揮し、バックグラウンド処理やバッテリー駆動時に適した特性を有することが確認されました。

また、本稿では一般的に語られる「NPUは省電力で優れている」という評価を踏まえつつも、処理対象となるモデル構造との相性に着目しました。たとえば、音声認識や分類などの軽量なタスクでは、NPUは平均6W未満というきわめて低い消費電力で高い電力効率を発揮し、バッテリー駆動環境やバックグラウンド処理に適した実行ユニットであることが確認されています。一方で、SLMのように中間テンソルが大きく、非局所的なメモリアクセスが頻発するモデルでは、NPUのSRAM構造やDMA転送が性能の制約要因となる場合があり、生成遅延やW/tokenの上昇につながるケースも見られました。

このように、NPUには短時間の軽量処理における高効率という明確な強みがある一方で、モデルの構造やタスク特性に応じて最適な処理ユニットを使い分ける必要があることが、定量データにより裏付けられました。こうした知見は、xPU（CPU/GPU/NPUなどを組み合わせたヘテロジニアス・コンピューティング）をアプリケーション側でどう設計・活用するかが、今後のAI PC実装において極めて重要な論点であることを示しています。

こうしたxPUの活用は、ユーザー側の選択に任せるものではなく、アプリケーションレイヤーが主導して行うべき最適化設計です。この認識に基づき、K-kaleidoではAIモデルをアプリケーションから分離し、GPU/NPUの選択とモデル切り替えを集約制御する「Local AI Model Hub」構成を開発しました。このHubを介することで、複数アプリケーション間でのモデル共有や、メモリ効率の最適化、ユーザーごとの利用シーンに応じた自律的なxPU制御が可能になります。

さらに、K-kaleidoではこの基盤を中心としたAI PC向けアプリストア「AI Edge Hub」の立ち上げを2025年9月に予定しています。Whisperを用いたオフライン通訳、VS Codeとの連携によるコード補完、業務文書の要約生成など、「現場で本当に使えるローカルAIアプリケーション」の展開を進めており、セキュアかつ持続可能なAI体験を現実のものとするUX構築に取り組んでいます(https://k-kaleido.com)。

AI PCは、単なるNPU搭載PCではなく、企業のAI活用における「クラウド依存を超えた自律的な選択肢」として再定義されつつあります。本稿がその可能性を判断し、自社にとっての戦略的価値を見出す一助となることを願っています。

Part.1 : はじめに — AIを、企業の手元に。

生成AIの進化とともに、AIの活用はもはや特別なものではなく、あらゆる業務を効率的に、そして創造的に進めるための前提条件となりつつあります。その多くは、クラウド上で提供されるSaaS (Software-as-a-Service)やAPI (Application Programming Interface)を通じて実現されていますが、一方で企業や自治体の現場からは、次のような声も聞こえてきます。

「会議の録音データをクラウドにアップロードしてよいのか?」
「機密性の高い処理は社内で閉じたい」
「クラウドAIのコストが、年々予測しづらくなっている」

つまり今、AIの「利便性」だけでなく、「制御可能性」や「持続可能性」が問われるフェーズに入ってきたと言えるでしょう。
そのような中で、「AI PC」という新しい選択肢が登場してから、1年以上が経過しました。
専用のNPU (Neural Processing Unit)を搭載し、PC単体でAI推論を実行できるという構想は、さまざまな場面で紹介され、今も進化を続けています。
ただ、生成AIの実務活用が急速に広がる一方で、現場からはこんな声も少なくありません。

「AI PC って、実際に何ができるの?」
「便利そうだけど、自社でどう活かせばいいのかわからない」
「クラウドだけに頼らずAIを活用できたら理想的だけど…」

つまり、技術や構想が少しずつ浸透しつつある一方で、それを業務にどう“引き寄せるか”という実感値がまだ定まりきっていない。
それが、現在のリアルな状況ではないでしょうか。
本ホワイトペーパーでは、インテル® Core™ Ultra 7 プロセッサー (開発コードネーム: Lunar Lake)を搭載したHP Elitebook X G1iを用い、AI PCという新しい選択肢の可能性を検証します。
この端末の進化が、企業に広がりつつあるAI活用の利便性を支える一方で、同時に課題とされるコストや機密性の問題に対して現実的な解となりうるのか。本書ではその点を検証し、クラウドAIとローカルAIを組み合わせた“ハイブリッドAI”の有効性を探っていきます。
また、開発者にとってもAI PCが新たなビジネスチャンスとなる可能性を秘めていること、そして企業と開発者をつなぐ新たなエコシステムが動き始めていることにも、あわせて触れていきます。

なぜ「ローカルAI (AI PC)」が必要なのか?

— クラウド依存が抱えるリスクと、ローカル実行という現実的な選択肢 —

AIの活用は、社内の会議や接客、資料作成から、開発や設計といった専門業務まで、企業活動のあらゆる現場に広がっています。こうしたAIを実行する場として、間違いなく主役はクラウドです。高性能なGPUや最先端の大規模モデルを柔軟に利用できるクラウド環境は、多くの企業にとってAI導入のハードルを下げてくれた立役者と言えます。

しかし、その一方で「クラウド前提のAI活用」にはいくつか見過ごせない課題もあります。特に、セキュリティの制御、コストの予測性、ネットワーク依存といった面では、現場での運用に不安を感じる企業も少なくありません。

こうした背景から、いま注目されているのが「ローカルAI」、すなわちAI処理をPC端末側で直接実行する仕組みです。以下では、クラウド依存がもたらす構造的リスクと、それに対するローカルAIの役割について、3つの視点から解説します。

1. セキュリティ：クラウドでは守りきれない領域がある

クラウドサービスは、現代のITインフラにおいて欠かせない存在となりました。

初期投資の回避、スケーラビリティ、運用負荷の軽減などのメリットにより、多くの企業がオンプレミスからクラウドへと移行を進めてきました。特にITリソースが限られる中堅・中小企業にとって、クラウドは現実的かつ合理的な選択肢となっています。クラウドベンダー側のセキュリティ対策も進化を続けており、データセンターは物理・論理の両面から厳重に保護され、通信の暗号化や多層的なアクセス制御、インシデント対応体制など、堅牢な仕組みが整備されています。懸念であったセキュリティレベルも上がってきているという声も聞かれます。

一方で、クラウドへの移行を見送っている企業や業界も依然として存在します。自治体や公共機関、金融機関、製造業などの分野では、法制度・業界ガイドライン・取引先要件・事業継続計画（BCP）などの観点から、クラウド利用に慎重な姿勢を維持しているケースも少なくありません。

安全性は高まったという声と、やはり慎重になる企業がどうして存在するのか。それは、いくつかの理由によると考えられます。

一つ目は、クラウドサービス上のセキュリティの責任範囲です。たとえばIaaS（Infrastructure as a Service）では、クラウド事業者の責任はインフラ基盤に限定されており、その上に構築されるOSやミドルウェア、アプリケーション、ユーザー管理、アクセス制御などはユーザー企業側の責任領域です。現実に、情報漏洩などのセキュリティインシデントの多くが、設定ミスや運用上の不備によって引き起こされています。近年では、WizやSysdigといったクラウドセキュリティプラットフォームの導入も進んでいますが、これらはあくまでユーザー側での設定や運用管理の支援を行うものであり、根本的な責任構造を変えるものではありません。ユーザー企業の対処能力によって、セキュリティレベルは大きく変わるわけです。

二つ目は、クラウド環境において、自社データが物理的にどこに保管されているのかを完全に把握・管理することは難しいという特性もあります。ベンダーによってはリージョン指定が可能な場合もありますが、「物理的に見えない」「監視しきれない」というクラウド特有の構造に対する不安要素も存在するといえます。これをリスクだと感じるかどうかは、企業ポリシーに準ずるわけで判断は分かれるでしょう。

クラウドセキュリティについては、この責任分界点をきちんと把握し、クラウドサービスプロバイダだけでなく、ユーザー企業においても適切な監視運用が求められる、という点を再認識すべきでしょう。

こうした構造的な課題が残るなか、近年では生成AIの業務活用が急速に現実味を帯びており、クラウドを利用する価値に「AIによる業務効率化」という新たな要素が加わっています。クラウド導入を進める企業にとっては、AIの利便性が導入推進の強力な後押しとなっていますが、一方で慎重な企業にとっては、「クラウドが前提となる限りAI導入は困難」という矛盾が突きつけられているのも事実です。

こうして、生成AIの登場がクラウド活用の判断にさらなる分岐を生んでいる状況下で、改めて注目されているのが、PC上でAIモデルを実行する「AI PC」のようなローカル処理環境です。

クラウドを経由せず、PCのNPUやGPUで処理を完結できる構成であれば、

- 機密性の高い情報の外部流出リスクを極小化し、
- 通信トラフィックやネットワーク統制の負荷を軽減し、
- 「誰が・どこで・何を処理したか」の可視性も高められる

といったセキュリティ面での利点が期待されます。

しかし、AI PCの価値はセキュリティ対策にとどまりません。生成AIの利用が本格化するほど、クラウド依存による従量課金の増加や処理レイテンシーといった実務上のボトルネックが深刻化しつつあります。さらに、クラウドを利用できない企業では、「AIを使いたくても使えない」という構造的な不公平感が現場に根付いてきています。こうした状況において、AI PCは「コスト」「応答速度」「情報統制」の三要素をバランスよく両立し、持続可能なAI活用を実現するための現実解として期待されています。

もちろん、ローカルAIには性能やモデルサイズといった制約もありますが、「すべてをクラウドで処理する必要はない」という選択肢を与えてくれる点が重要です。クラウドとローカルの役割を適切に切り分け、業務内容やリスクレベルに応じて処理場所を柔軟に選ぶ。このようなハイブリッド運用こそが、コストとセキュリティの最適解を導く鍵となると考えられます。そして、これまでクラウドを「使わない」ことを前提にシステムを構築してきた企業にとっても、AI PCはセキュリティを確保しながらAIを活用するための、現実的な第一歩になるかもしれません。AI PCは、さまざまなIT制約や社内ポリシーを抱える組織にとって、“無理なく始められる新しい選択肢”として、AI活用の幅を着実に広げつつあります。

2. コストとサステナビリティ：青天井のクラウドコスト、見えにくい環境負荷

クラウドは「使った分だけ払う」仕組みで、初期投資を抑えやすいというメリットがあります。とくにAI系サービスでも、API経由で必要なときに処理を呼び出せるのは非常に便利です。

しかしその一方で、使えば使うほどコストがかさむという側面も無視できません。たとえば音声認識や要約、翻訳などの生成AI処理を日常的に活用する場合、従量課金の単価は決して安くなく、月々の費用が想定以上に膨らむこともあります。SaaSのように月額固定で予算化しやすい仕組みと違い、APIベースのAI活用では予算見通しが立てづらく、経理・経営面での不安要素となりがちです。

また、AIは導入後すぐに効果が出るものではなく、現場での試行錯誤を通じて徐々に業務にフィットさせていくプロセスが重要です。ところが、従量課金型の仕組みではその試行錯誤にもコストが発生するため、「効果が見えないうちは使いにくい」というジレンマも生まれます。企業としては、AI導入の勢いにブレーキをかけつつ、ROIを意識した賢い運用が求められるフェーズに入りつつあるのです。

もう一つの視点が「環境負荷」、そしてそれに伴う「将来的なコスト」です。クラウドサービスを提供している巨大なデータセンター群が大量の電力と冷却資源、土地というリソースを大量に消費している、という現実です。国際エネルギー機関（IEA）の予測では、世界のデータセンターの電力消費量は2030年までに945TWhに達する見込みで、これは日本の総発電量に匹敵しますharvardmagazine.com+13iea.org+13deloitte.com+13。さらに、北米ではデータセンターの急増により家庭向け電気代が2040年に年間150〜450ドル上昇するという試算も出ていますhttps://www.businessinsider.com/big-tech-secret-energy-deals-utility-bills-cost-consumers-2025-3?utm_source=chatgpt.com。これは家庭向けの数字ではありますが、電力単価の上昇が法人向け契約にも波及する可能性は高く、企業にとっても運用コストとして跳ね返ってくるリスクを示唆しています。

(注釈: 家庭の平均年間消費電力10,000kWhを前提とすれば、単価上昇は+0.015〜0.045 USD/kWh程度 → 企業が年間100万kWh使用する場合、15,000〜45,000 USDの追加コストに相当します)

さらに、マッキンゼーは今後データセンターへの設備投資が2030年までに6.7兆ドルに達するとの予測も出しており、資本コストの増加もクラウド料金に転嫁されていく可能性があると考えられます time.com+15mckinsey.com+15cbsnews.com+15。

AIの常用が進むなかで、クラウド依存による「予算予測の難しさ」「外的要因による変動リスク」「環境コストの増加」などが顕在化しつつあります。こうした不確実性に備える意味でも、クラウドを基軸としつつ、AI PCのようなローカル実行環境を併用することで、コストや運用負荷に対するブレーキを持つ選択肢を確保することが可能になります。

特にAI PCは、初期段階でCPU/GPU/NPUを備えた端末を調達すれば、それ以降のAI実行において追加の従量課金（OPEX）が発生しないという構造的な強みがあります。さらに、こうしたAI処理能力は、すでに業務用に配備されたPC端末に内在しているケースも多く、あらためて大きなCAPEXを要するわけではありません。

たとえば企業内に配備された1,000台のPC端末が、それぞれ微力ながらもAI処理能力を備えていれば、これらを分散的に活用することでクラウドへの依存度を一定程度低減できます。このような「未活用リソースの再定義」は、AI対応のエッジ端末としての価値を引き出すものであり、TCO最適化やサステナビリティ戦略の一環として十分に検討に値するアプローチといえるでしょう。

3. 業務継続性：
ネットワークに依存しない“自立性”の価値

クラウドを前提としたAI活用は、ネットワーク接続が「生命線」となります。しかし、現実のビジネス環境では、常に最良の条件（高速・安定な通信環境、低レイテンシー、クラウドAPIの可用性）が整っているとは限りません。たとえば、会議中にリアルタイム文字起こしを試みたが、APIの遅延で使い物にならなかった、あるいは出張先で翻訳が必要だったのに、ネットワークの不安定さで動かなかったといった事例は、決して珍しいものではありません。AIを業務に組み込んでいく上では、「すべてがうまくいく前提（ベストケース）」で構築するのではなく、「一部がうまくいかなくても動き続ける（ワーストケース）設計」が、より広く活用するためには不可欠です。これは、単なる利便性の問題ではなく、業務品質やビジネス継続性（BCP: Business Continuity Plan）に直結する根本的な設計思想です。

AIの処理には、たとえば議事録の文字起こしや翻訳のような比較的シンプルな処理から、大規模な生成系モデルによる複雑な推論まで、さまざまなレベルがあります。

これまで、AI処理すべてをクラウドに頼る構成をとらざるを得ませんでした。それは、ローカルのPCにはAIを動かすという発想そのものがなかったからです。

しかしいま、AI PCと呼ばれる新世代の端末は、一定レベルのAI処理をPC単体で行える性能を備え始めています。もちろん、現時点ではすべてのAI処理がローカルで完結するわけではありません。けれど、「できることはローカルでやる」というシンプルな設計思想は、業務にAIを根付かせていくうえで、極めて現実的かつ有効なアプローチです。

たとえば、

- 通信環境が不安定な出張先でもリアルタイム文字起こしができる
- セキュアな社内ネットワーク内でも生成AIが使える
- オフライン環境下であっても最低限のAIサポートが機能する

こうした「クラウドが使えない場面でも、最低限のAI機能が継続できる」状態を確保しておくことは、現場でAIを安心して定着・活用していくための、重要な土台となります。さらに将来的には、PCに搭載されるAI処理能力（GPUやNPU）の進化によって、より高度な処理をローカルでこなすことも可能になっていくでしょう。

私たちは、クラウドとローカルを組み合わせたハイブリッドAI活用という選択肢を、現実解として育てていくことが、AIの持続的な業務活用において欠かせないと考えています。

観点	クラウドAI	ローカルAI（AI PC）
実行場所	データセンター（遠隔）	ユーザーの手元PC内で実行
通信依存性	高：常時インターネット接続が必要	低：基本的な処理はオフラインで完結可能
セキュリティ管理	一部クラウド事業者依存＋設定ミスによるリスク	データは端末内完結、漏洩リスクを最小化
コスト構造	従量課金型（使用量に応じて変動）	固定費型（端末に含まれるため、使用による追加費用なし）
スケーラビリティ	高：計算リソースを一時的に大量確保可能	端末スペックに依存（用途ごとに十分な能力があれば可）
データ主権・制御性	部分的（データ処理はクラウド側に委託）	完全（ユーザー自身がデータ処理と保持を制御）
BCP対応力（障害耐性）	ネット障害時に機能停止	ローカル動作により冗長性あり
導入・運用負荷	小（設定だけで利用可能）	やや高（機器準備・アプリ導入が必要）
代表的な用途	ChatGPT、画像生成、クラウドAPI連携など	会議議事録生成、音声文字起こし、要約、翻訳、プログラミング支援など

AI PCとは?従来のPCやCopilot+ PCとの違いは?

AI PCという言葉を耳にする機会が増えてきましたが、「結局なにが違うの?」という声も少なくありません。このセクションでは、従来のPCや最近話題のCopilot+ PCと比較しながら、「AI PCとは何か」を丁寧に整理します。

AI PCとは?

端的に言えば、AI PCとは「NPU（AI処理専用装置: Neural Processing Unit）」というAI処理に特化した専用回路を搭載したPCです。現時点でインテルが定義するAI PCの基準に最も合致するのが、「インテル® Core™ Ultra プロセッサーを搭載したPC」です。従来のPCは、主にCPU（中央演算処理装置: Central Processing Unit）とGPU（画像処理装置: Graphics Processing Unit）によって処理を行っていました。AIの登場により、「AI専用の処理を、いかに効率的に、かつ低消費電力で行うか」が重要な課題となっています。そこで登場したのがNPUです。

xPU時代の設計思想：ヘテロジニアス・コンピューティング

PCの頭脳であるCPUはあらゆる処理をこなす万能型ですが、電力効率は決して最適とは言えません。特にノートPCのようなモバイル環境では、バッテリー持ちと処理性能のバランスが重要になります。
そこでインテルは、「頻繁に行う処理は、最も効率的に処理できる専用回路に任せよう」というヘテロジニアス・コンピューティング（xPU）の思想を導入しました。

- CPU：あらゆる処理の司令塔（汎用的）
- GPU：重いAI推論や画像処理に最適（性能重視）
- NPU：音声認識やリアルタイム翻訳などのAI処理を、超低消費電力で実行（省電力重視）

つまり、「適材適所」で処理を分担することで、性能と電力効率の両立を目指すのがAI PCの根本的な設計思想です。

Copilot+ PCとの違いは?

Copilot+ PCはマイクロソフトが定めるAI体験を提供するPCのプレミアムブランドであり、

- 40TOPS（1秒当たり40兆回）以上の処理性能を持つNPUの搭載
- 16GB以上のDDR5 ／ LPDDR5(X)規格メモリ
- 256GB以上のSSD ／ UFSストレージ
- 「Copilot」キーの搭載

が要件とされ、より厳格な仕様を満たした一群のPCに付けられる称号です。
したがって、Copilot+ PCはAI PCの中に含まれる、より高機能で体験重視のプレミアムカテゴリと言えます。

AI PCを支える設計思想：Lunar Lakeの中身とAI処理の最適化

2024年に発表された「インテル® Core™ Ultra プロセッサー（開発コード名:Lunar Lake）」は、AI PCの中核を担う次世代プロセッサーとして、Copilot+ PCの要件も満たす高性能かつ低消費電力なプラットフォームです。この章では、その内部アーキテクチャと、なぜNPUとGPUを併載する構成が現実的なのかを、AI処理の実情に即して解説します。

なぜ“複数の頭脳”が必要なのか？：xPU時代の設計思想

AI PCは「万能なCPUで全てを処理する」時代から脱却し、AI処理を適切なコンピューティングユニットで実行することで、最適なユーザー体験を実現することがコンセプトです。Lunar Lakeでは、以下のように機能分担された3つのプロセッサーが協調動作する「ヘテロジニアス・コンピューティング」が採用されています。

Lunar Lakeに搭載された3種のプロセッサー（CPU/GPU/NPU）の役割

プロセッサー	主な用途	特徴
CPU（Lion Cove ／ Skymont）	汎用処理、制御系	高汎用性。だがAI処理は非効率
GPU（Xe2 + XMX）	高負荷な推論処理、画像生成	最大67TOPS。高スループット
NPU（Neural Processing Unit 4）	省電力なAI常時処理	最大48TOPS。低レイテンシー・低消費電力

この分業体制により、シーンに応じて「処理性能を重視する場面」「バッテリーを重視する場面」「ユーザーが気づかぬ裏側で動作するAI機能」などを柔軟に切り替えることが可能になります。

メモリ構成とアクセラレータの“相性”：性能の鍵はここにある

GPUとNPUの性能差を語るうえで、見逃せないのが「メモリアクセスの仕組み」です。

- GPU（Xe2アーキテクチャ）は、高速な並列演算とメインメモリ共有アクセスを活かし、大規模なAIモデルや長文推論に適します。さらに、XMX（Xe Matrix eXtension）エンジンにより、行列演算のスループットが大幅に向上。キャッシュ階層（Last-Level Cache）もCPUと共有しており、データ再利用性にも優れます。
- NPU は、処理に必要な中間テンソルを主に内蔵SRAMに保持しながら高速処理する構造です。このSRAMとメインメモリ間のデータ転送はDMA (Direct Memory Access)であり、GPUのようにキャッシュ構造が使えないため、NPUからアクセスできるメモリ空間に制約があります。そのため、処理サイズが一定以下で完結する軽量モデル（短文翻訳、感情分析など）では極めて高効率。一方で、トークン数が多くなると中間データが溢れ、処理停止や性能劣化の原因になる可能性があります。

このように、同じ「AI推論」でも、処理対象によって最適なアクセラレータは異なります。AI PCはその違いを前提に設計されており、Lunar Lakeはまさにその思想をハードウェアで体現しています。

実利用に向けた進化：薄型軽量ノートパソコンでもAIが当たり前の体験へ

従来のノートPCでは、GPUをAI処理に活用しようとすると電力・発熱の制約、またバッテリー駆動時間への影響がネックでした。しかしLunar Lakeでは、以下のような構成により、日常的にAIを使うというユーザー体験が実現可能になります：

- 省電力NPUで、ZoomやTeamsなどの会議ツールの文字起こしを常時ON
- 高性能GPUで、PowerPointでの画像生成支援や要約応答を即座に実行
- CPUとの連携で、処理の中断や切り替えをシームレスに実行

つまり、いつでもどこでもAIが自然に動く体験を支えるインフラとして、AI PCはこれまでの「高性能＝高性能CPU」という常識を転換し、最適ユニット活用によるスマートな高性能へと進化させたのです。

この章で示したように、Lunar Lakeは単なるスペックの進化ではなく、「用途ごとに処理資源を適切に割り当てる」という新しい思想のもとに構築されたプラットフォームです。次章では、実際にこのGPU/NPUそれぞれを用いたAIタスクの性能や応答性を比較し、その設計思想がどのようにユーザー体験に結びついているのかを、定量的に評価します。

Part 2 : AI PCアプリケーション検証

検証の目的と前提条件

これまで、生成AIはクラウド上で動かすのが常識とされてきました。しかし、AI PCの登場により、生成AIを「手元」で動かすという新たな選択肢が現実味を帯びています。では、ローカルAIは実務に耐えうるのか? この問いに答えるため、私たちは最新のインテル® Core™ Ultra 7 プロセッサー（開発コード名：Lunar Lake）を搭載したHP Elitebook X G1iを用い、自然言語処理とコード生成という業務的なユースケースに即して、処理性能・応答品質・電力効率を多角的に評価しました。

主なスペックは下記のとおりです

- CPU: インテル® Core™ Ultra 7 プロセッサー
- Memory: LPDDR5X 32GB (オンパッケージ)
- Graphics: インテル® Arc™ グラフィックス (XMXエンジン搭載, 67TOPS)
- NPU: Intel NPU 4 (47TOPS)
- Storage: 1TB SSD
- OS: Windows 11 Pro

検証は「最適なパフォーマンス」モードで、バックグラウンドアプリを最小限に抑えた状態で実施し、OpenVINO RuntimeおよびPyTorchベースの変換モデルを用いてローカル実行を行いました。

ローカルSLMは実用に足るか?
—出力品質と電力効率の単体評価

検証の第一ステップでは、「クラウドLLM(Large Language Model: 大規模言語モデル)に頼らずとも、ローカルSLM (Small Language Model: 小規模言語モデル)で業務に十分有用な応答が得られるのか?」という観点から、軽量モデルの回答品質と電力効率を軸に評価を行いました。特に、GPUおよびNPUといった処理エンジンごとの実行特性に着目し、INT4(4bit整数型)量子化(AIモデルの軽量化)済みのSLMを実業務に近い複数タスクで比較検証しています。表に今回評価したモデルの一覧を示します。

モデル名	特徴	主な用途	評価実行場所
Elyza Llama3 JP 8B	日本語特化、8B、軽量SLM	自然言語・コード両対応	GPU / NPU
DeepSeek R1 Qwen 14B	汎用モデル、14B、やや大型	自然言語処理	GPU
Starcoder2 15B	コード特化、15B	プログラミング支援	GPU

- https://huggingface.co/Elyza/Llama-3-Elyza-JP-8B
- https://huggingface.co/cyberagent/DeepSeek-R1-Distill-Qwen-14B-Japanese
- https://huggingface.co/bigcode/starcoder2-15b

測定は、業務向け自然言語処理のシナリオ10パターン、プログラミングコード系のシナリオ10パターンを実施しました。なお、Starcoder2はGitHub上のコードを基に事前学習された、コード生成特化型のSLMであるため、プログラミングコード系の処理のみで評価します。

評価の進め方

上記のモデルと環境をもとに、本検証では以下2つの視点から評価を行いました。各セクションでは、これらの評価の背景・観察ポイント・得られた知見について詳述していきます。

1. 出力品質の比較：INT4量子化モデルの応答評価

PC上で動作するAIモデルとしての有用性を見極めるため、INT4（4bit整数型）に量子化されたSLM（Small Language Model）を使い、自然言語処理およびコード処理における出力品質を評価しました。

- ●自然言語処理タスク（10パターン）に対して：
  - ○使用モデル：
    - ・Llama-3 Elyza JP 8B（GPU ／ NPU実行）
    - ・DeepSeek R1 Qwen 14B（GPU実行）
  - ○評価方法：
    - ・同一プロンプトに対する各モデルの出力を比較
    - ・GPT-4oによる参照出力と比較し、Cosine類似度により人間にとっての意味の近さを数値化（0〜1）
- ●コード生成タスク（10パターン）に対して：
  - ○使用モデル：
    - ・上記に加えて Starcoder2 15B（GPU実行）
  - ○評価方法：
    - ・同一プロンプトに対する各モデルのコード出力を比較し、GPT-4oとの出力傾向を対照分析

2. 処理性能と電力効率の評価：実行エンジン別ベンチマーク

出力品質に加えて、「処理スピード」「消費電力」「電力効率」の観点から、GPU ／ NPU上で動作するINT4量子化SLMの性能を定量的に比較しました。

- ●測定対象：
  - ○FTL: First Token Latency（最初の応答が返るまでの時間でユーザー体感に直結）
  - ○トークン生成速度（tokens/sec）
  - ○処理全体の消費電力（Power Consumption）
  - ○1トークンあたりの消費電力（W/token）
  - ○平均電力（W）

これらの評価項目を通じて、AI PCにおけるモデルと処理ユニットの最適な組み合わせを分析し、ユースケースに応じた最適化の指針を導き出します。

以下では、2つの評価軸それぞれについて、詳細な検証結果と考察を順に示していきます。まずは「出力品質の比較」から、その結果と知見を紹介します。

1. 出力品質の比較：INT4量子化モデルの応答評価

前述の評価方針に従い、出力品質に関する評価・検証を行いました。以下では、その結果についてまとめます。

まず、自然言語処理タスクでは、短文要約、FAQ自動生成、ビジネス文書の英訳といった定型的な業務ユースケースを中心に検証を実施しました。L3 Elyzaモデルは、情報の要約力や構文の簡潔さ、文意の伝達力において安定した出力を示し、GPT-4oを参照出力とした比較でも、十分に実務転用可能な水準に達していることが確認されました。特に、冗長性を抑えた応答傾向や、日本語処理に特化した構成によって、応答の自然さと効率が両立していた点は特筆すべき結果です。

また、INT4量子化された状態においても、L3 Elyzaは構成力や論理展開が求められる長文生成タスクで大きな破綻を起こすことなく、一定の完成度を維持しており、軽微な編集を前提とすれば業務文書の下書き用途として有効に活用できる水準に達しています。一方、14BのDeepSeek R1は、文脈理解や推論においてやや深みのある出力を示す傾向も見られましたが、モデル規模の差がそのまま実用性の差に直結するわけではないことが示唆されました。

コード処理領域では、L3 ElyzaモデルはPython、SQL、JavaScriptなどの多様な言語に対して、構文の正確性、命名規則の整合性、関数構造の明瞭さといった観点から、安定した出力品質を示しました。短い関数生成やテンプレート生成といった軽量タスクでは、冗長性の少ないL3 Elyzaの出力のほうが、理解や修正がしやすいという点で現場的な実用性を持つケースも多く確認されました。

加えて、コード特化型の15BモデルであるStarcoder2との比較では、コメント生成やエラーハンドリングの丁寧さにおいてStarcoder2が優位な場面も見られた一方で、L3 Elyzaは構文の整合性や応答の一貫性において高い信頼性を保っており、ローカル実行環境での軽量性と出力の安定性を両立するモデルとして、現実的な選択肢になり得ることが確認されました。

さらに、NPU上での制約がある環境においても、L3 Elyzaは設計指針や処理の意図に関する簡潔な提案を出力できており、たとえ出力全体が完結しなくとも「補助的なアウトライン生成」や「レビュー支援」といった用途で活用できる可能性が見出されています。

INT4量子化を施したSLMにおいても、応答の文法構造や出力の自然さは維持されており、各種業務タスクに対して十分に実用的な品質を保っていることが確認できました。特にL3 Elyzaモデルは、日本語に特化した設計と8Bという構成のバランスにより、自然な表現生成と応答の安定性の両立を実現しています。さらに、分類系タスクではNPU上の処理でGPUを上回る性能を示すなど、モデルの構造や用途に応じたアーキテクチャ選定の重要性も示唆されました。これらの結果から、INT4量子化SLMは、ローカルAIの実装において、導入可能性と実用性の両面で有力な選択肢となります。

2. 処理性能と電力効率の評価:実行エンジン別ベンチマーク

前章では、Elyza Llama3 Japanese 8Bを中心とするINT4量子化SLMが、自然言語処理・コード生成ともに実務に耐えうる出力品質を備えていることを確認しました。しかし、出力品質が十分であっても、それをどの処理エンジンで実行すれば、速度や消費電力の面でも実用的かどうかは、別の重要な検討課題となります。本章ではこの点に着目し、GPUとNPUという異なる実行ユニットにおける処理性能と電力効率を定量的に比較し、ユースケースに応じた実行戦略の最適解を探ります。

具体的には、GPUとNPUの違いが生成速度や消費電力、さらに1トークンあたりの電力効率にどのような影響を与えるかを軸に検証を進めます。あわせて、モデルサイズ（軽量なL3 Elyzaと、中量のDSR1・SC2）による傾向の違いにも注目し、AI PC上での実行パターンを導出していきます。

自然言語処理：応答性と電力効率の両立は可能か

自然言語処理タスクにおいては、L3 ElyzaをGPU上で実行した構成が、生成速度・応答品質・電力効率のいずれにおいても最もバランスが良く、実用面で高い適性を示しました。短文から長文まで、120～160 tokens/secのスループットを安定して維持しながら、平均電力は12～20Wに収まり、日常業務での常用に十分耐える実行環境であることが確認されました。

対照的に、NPU上での実行は、短文要約や翻訳といった軽量タスクにおいて、消費電力の低さという点で一定の優位性を示しました。ただし、First Token Latency（FTL）は最大で約1.8秒とやや遅延があり、体感レスポンスの面ではGPUに比べて不利な傾向が見られました。加えて、長文生成のように出力量が多い処理では、NPUの生成速度は50 tokens/secを下回るケースが多く、応答性の観点で明確な制約が浮き彫りとなりました。

一方、中量モデルのDSR1をGPUで実行した場合でも、100 tokens/secを超える生成速度を維持しており、モデルサイズが大きくなった場合でもGPUは一定のスループットを保ちやすいという特性が確認されています。

以上の結果から、自然言語処理における処理ユニットの選定は、生成速度、出力品質、消費電力という三要素のバランスによって左右されることが明らかになりました。特にL3 ElyzaをGPU上で動作させた構成は、簡潔な応答、十分な速度、低い電力負荷という三点を同時に満たしており、ローカルAIの基盤として信頼性の高い選択肢となります。一方で、NPUはバッテリー駆動環境やバックグラウンド処理など、軽量タスクを前提とした場面において、限定的ながら効果的に活用できるユニットと位置づけられます。

コード処理：生成とレビュー、どちらを重視するか

コード生成においても、L3 Elyzaは構文の正確さと明快な関数構造を維持し、GPU上での安定性に優れた出力を記録しました。SC2はコメント補足や例外処理の表現が豊かである一方、出力の冗長さや構造の一貫性にばらつきがあり、生成よりレビュー補助や教育用途に向いている傾向が見られました。

NPUでのコード生成は、出力途中での打ち切りが散見され、長文応答には不向きです。ただし、変数命名の提案や処理構造の整理といった「レビュー支援」や「設計方針の補助出力」には「小粒で強い応答」が求められる場面に特化して活用できる可能性が見出されました。

生成速度の観点では、NPUは概ね40～50 tokens/secで頭打ちとなった一方、GPUではElyzaが常時100 tokens/sec超を維持。SC2はタスクによっては200 tokens/sec超も記録しましたが、生成過多によって要点が埋もれる傾向もあり、出力の質と量のバランスが求められる局面が多く見られました。

電力効率の観点から見る最適配置

NPUは、全タスクを通じて平均消費電力が6W未満に収まり、静音性や発熱制限、バッテリー駆動が求められるモバイル環境では非常に有効な実行ユニットです。特に、短時間で完結する分類処理や定型応答といった軽量タスクでは、その低消費電力性が際立ちました。

一方で、NPUは構造的にトークン生成のスループット（tokens/sec）が低く、長文出力などの処理では生成に時間がかかる傾向があります。その結果として、たとえ平均消費電力が小さくても、生成にかかる時間が長くなることで、処理単位あたりのエネルギー効率が相対的に低下するケースがあります。特に、バッテリー駆動を重視するユースケースでは、平均電力の低さだけでなく、「一定の成果（例えば1トークンの出力）を得るためにどれだけの電力が必要か」という視点での比較が重要になります。

こうした背景から、本検証では、1トークンを生成するのに必要な消費電力を示す指標として、W/token を導入しました。W/token は以下の式で定義されます。
W/token = 平均電力（W）÷ 出力トークン速度（tokens/sec）

この値が小さければ、 少ない電力で効率よく出力が得られる、つまり電力効率が高くなります。

GPU実行では、L3 Elyzaモデルで12～20Wの平均電力で安定した生成速度を維持しました。DSR1やSC2など中量モデルでは処理スループットがさらに向上する一方、1000～1800mWhのトータル電力消費に達するタスクも存在しました。ただし、トークン生成速度の速さが功を奏し、W/tokenではNPUより優位な結果を残しています。

こうした違いの背景には、GPUに搭載されたXMX（Xe Matrix eXtension）の存在が大きく寄与しています。XMXは行列演算に特化した専用ユニットであり、Transformer型AIモデルが行う膨大な計算、たとえば「128個の単語ベクトル（各4096次元）間の関連性をすべて掛け合わせ、その重み付け結果をもとにベクトルを合成する」といった処理に対して、 高速かつ効率的に対応できます。

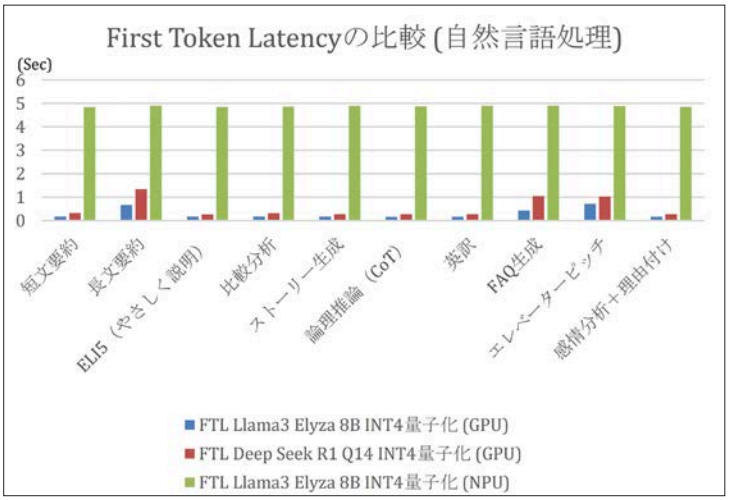
これらの処理は、AttentionやMLP（多層パーセプトロン）といったTransformerの中核機構で頻繁に実行され、XMXのように行列積（MatMul）や積和演算（FMA）に最適化された演算器が極めて有効です。

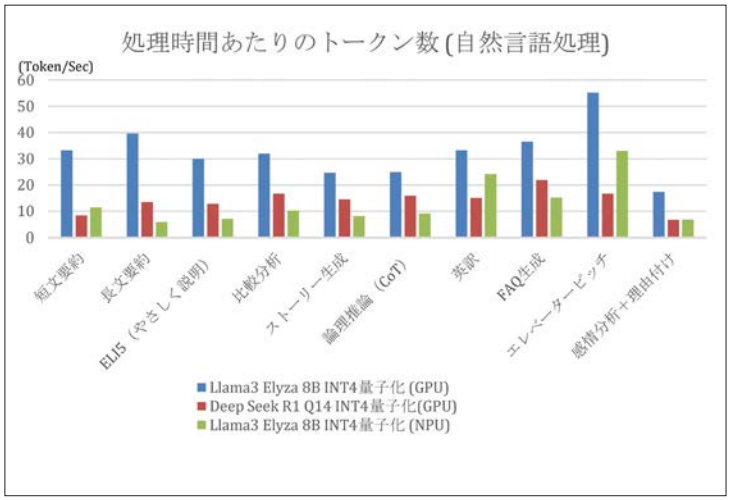
また、 GPUはメモリアクセスのアーキテクチャにおいてもNPUに対して有利です。 NPUが主にローカルSRAMとDRAM間をDMA転送によって処理するのに対し、GPUはDRAMおよびラストレベルキャッシュへの直接アクセスを持ち、オンダイのリングバスを介して高速転送が可能であるため、低レイテンシーかつ広帯域なデータ処理が可能となっています。

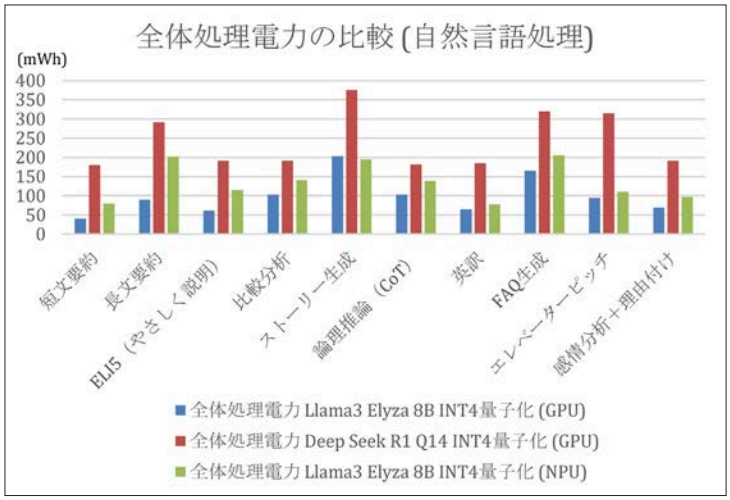
これらを踏まえた処理ユニットとモデルの最適な組み合わせは以下の通りです：

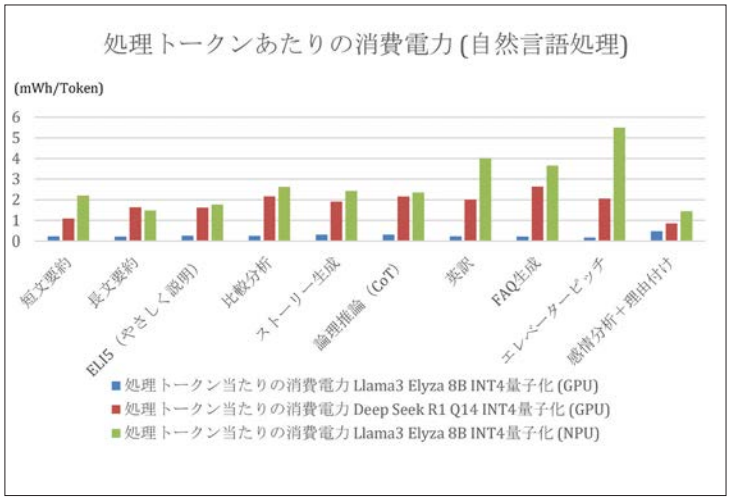
実行ユニット × モデル規模	適性シナリオ
NPU × 軽量モデル	常時実行、定型応答、レビュー補助
GPU × 軽量モデル	高速生成、日常業務アプリへの組込み
GPU × 中量モデル	説明付き出力、レビュー・学習支援、深い文脈理解

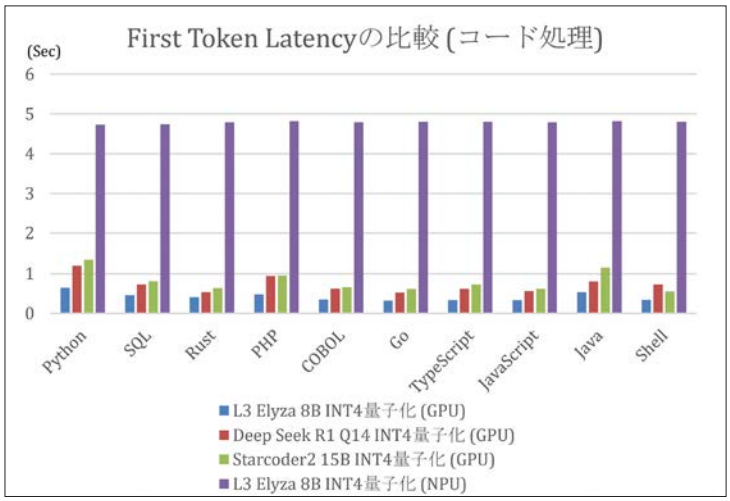
分類タスクや短文応答ではNPUの電力効率が活きる一方、生成ボリュームの多い処理ではGPUの演算・帯域性能が真価を発揮します。 つまり、処理密度（performance per watt）と電力効率の最適なバランスを見極めることが、AI PCを単なる高性能端末から、戦略的なAI活用基盤へと昇華させる鍵となります。

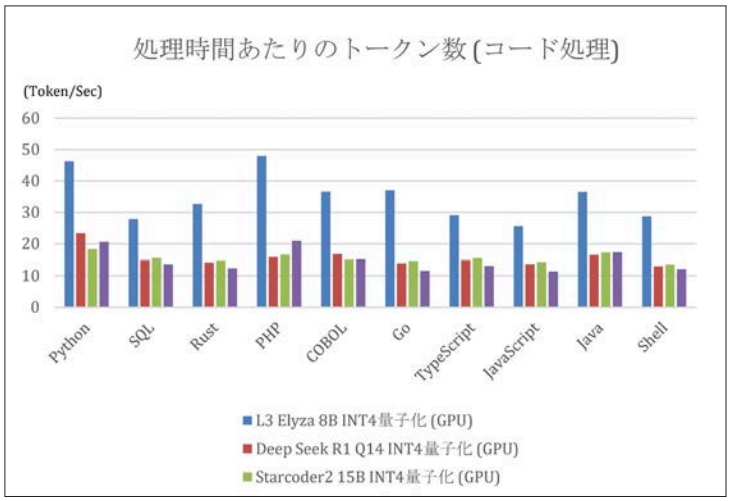


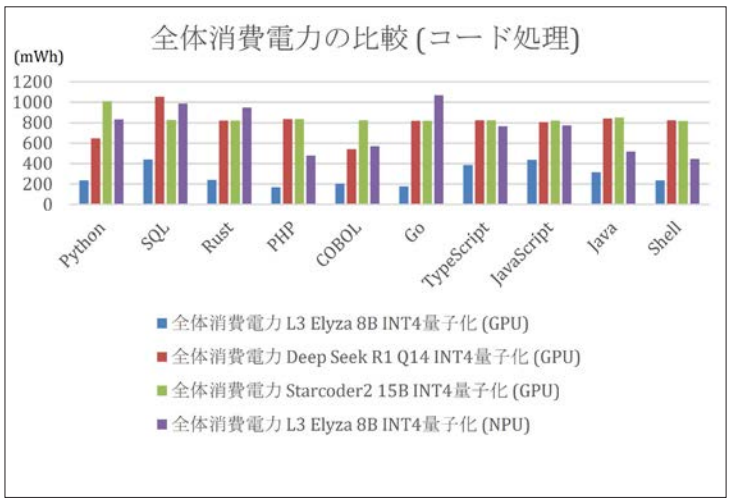


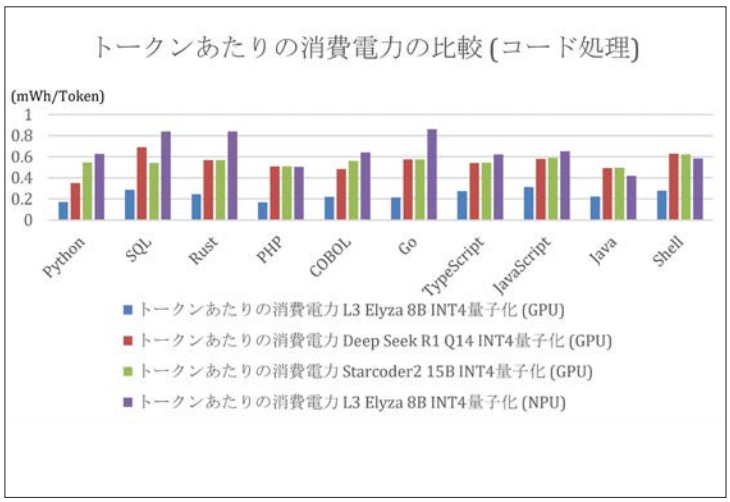

全体処理電力の比較 (自然言語処理)
(mWh)
400
350
300
250
200
150
100
50
0
短文要約
長文要約
ELI5 (やさしく説明)
比較分析
ストーリー生成
論理推論 (CoT)
英訳
FAQ生成
エレベーターピッチ
感情分析＋理由付け
全体処理電力 Llama3 Elyza 8B INT4量子化 (GPU)
全体処理電力 Deep Seek R1 Q14 INT4量子化 (GPU)
全体処理電力 Llama3 Elyza 8B INT4量子化 (NPU)


処理トークンあたりの消費電力 (自然言語処理)
(mWh/Token)
6
5
4
3
2
1
0
短文要約
長文要約
ELI5 (やさしく説明)
比較分析
ストーリー生成
論理推論 (CoT)
英訳
FAQ生成
エレベーターピッチ
感情分析＋理由付け
処理トークン当たりの消費電力 Llama3 Elyza 8B INT4量子化 (GPU)
処理トークン当たりの消費電力 Deep Seek R1 Q14 INT4量子化 (GPU)
処理トークン当たりの消費電力 Llama3 Elyza 8B INT4量子化 (NPU)


First Token Latencyの比較 (コード処理)
(Sec)
6
5
4
3
2
1
0
Python
SQL
Rust
PHP
COBOL
Go
TypeScript
JavaScript
Java
Shell
L3 Elyza 8B INT4量子化 (GPU)
Deep Seek R1 Q14 INT4量子化 (GPU)
Starcoder2 15B INT4量子化 (GPU)
L3 Elyza 8B INT4量子化 (NPU)


処理時間あたりのトークン数 (コード処理)
(Token/Sec)
60
50
40
30
20
10
0
Python
SQL
Rust
PHP
COBOL
Go
TypeScript
JavaScript
Java
Shell
L3 Elyza 8B INT4量子化 (GPU)
Deep Seek R1 Q14 INT4量子化 (GPU)
Starcoder2 15B INT4量子化 (GPU)


全体消費電力の比較 (コード処理)
(mWh)
1200
1000
800
600
400
200
0
Python
SQL
Rust
PHP
COBOL
Go
TypeScript
JavaScript
Java
Shell
全体消費電力 L3 Elyza 8B INT4量子化 (GPU)
全体消費電力 Deep Seek R1 Q14 INT4量子化 (GPU)
全体消費電力 Starcoder2 15B INT4量子化 (GPU)
全体消費電力 L3 Elyza 8B INT4量子化 (NPU)


トークンあたりの消費電力の比較 (コード処理)
(mWh/Token)
1
0.8
0.6
0.4
0.2
0
Python
SQL
Rust
PHP
COBOL
Go
TypeScript
JavaScript
Java
Shell
トークンあたりの消費電力 L3 Elyza 8B INT4量子化 (GPU)
トークンあたりの消費電力 Deep Seek R1 Q14 INT4量子化 (GPU)
トークンあたりの消費電力 Starcoder2 15B INT4量子化 (GPU)
トークンあたりの消費電力 L3 Elyza 8B INT4量子化 (NPU)

アプリケーション実現例：
ローカルAI時代に求められる「賢いアプリ」とは

ローカルAIを現実的に活用するためのアプローチとして、K-kaleidoが開発したアプリケーション群を紹介します。具体的な事例に入る前に、まずAI PCにおけるメモリ使用の課題について整理しておきます。

INT4量子化されたSmall Language Model（SLM）であっても、1モデルあたりおよそ20GB前後のメモリを消費します。このため、メインメモリが16GBのPCでは明らかに不足し、32GBの構成であっても複数モデルの同時利用は難しくなります。より軽量なモデルへの切り替えも可能ですが、本稿で検証したように、Llama 3ベースの8Bモデルでようやく業務転用に耐えうる水準であり、それ以下ではユーザー体験の著しい劣化が避けられません。つまり、限られたメモリ領域を競合させず、AIモデルが排他的に逐次利用できる仕組みが必要となります。

この課題に対して、K-kaleidoでは「Local AI Model Hub」という共通基盤を活用したアーキテクチャを採用しています。これは、AIモデルをアプリケーションから分離し、ローカルPC上でModel Hubが1つのモデルだけを常駐・管理する仕組みです。アプリケーションはHTTP経由でModel Hubにリクエストを送信し、応答を受け取る構成となっています。

この設計によって、次のようなメリットが得られます：

- 複数アプリケーションでのモデル共有が可能（排他的に切り替えて活用）
- 使用されていないモデルは都度アンロードされ、メモリ使用を最小化
- クラウドAPIと構造が類似しており、既存アプリケーションからの移行が容易
- 処理内容やデバイス状態（負荷、バッテリー状況）に応じてAIエンジンの選択が柔軟

この仕組みにより、Model HubはAI PC上の複数アプリケーションにとって「ローカルAIモデルの共有サーバ」として機能し、メモリ効率とモデル管理の最適化を同時に実現します。

実際のAI PCでは、処理内容に応じてNPU、GPU、CPUのいずれを使用するかを動的に選択してAI推論を実行しています。これからのAIアプリケーションに求められるのは、こうした実行基盤に追随し、ユーザー体験を最適化できる「賢いアプリ」です。

実装例：2つのローカルAIアプリケーション

本稿では、K-kaleidoが実装した以下の2つのアプリケーション例を紹介します：

- Whisperベースのオフライン通訳アプリ（Local AI Model Hub非使用）
- VS Code向けコード生成補助AIプラグイン（Local AI Model Hub使用）

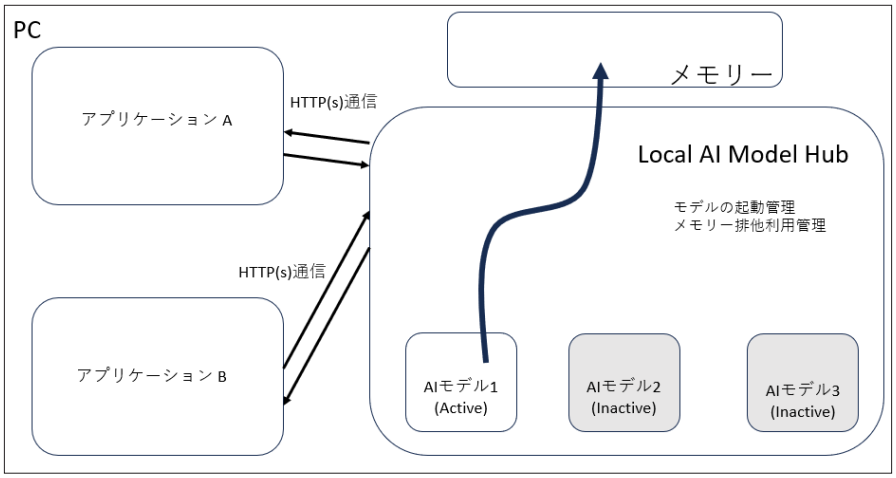
とくに後者は、Model Hubを介してモデルを切り替えながらリクエストを処理する構成を採用しており、1台のPC上で複数アプリケーションがAIモデルを共有する構成の実現可能性を実証しました。Model Hub上では、今回紹介した3種類のAIモデル（L3 Elyza、DeepSeek R1、Starcoder2）が切り替えられるようになっています。

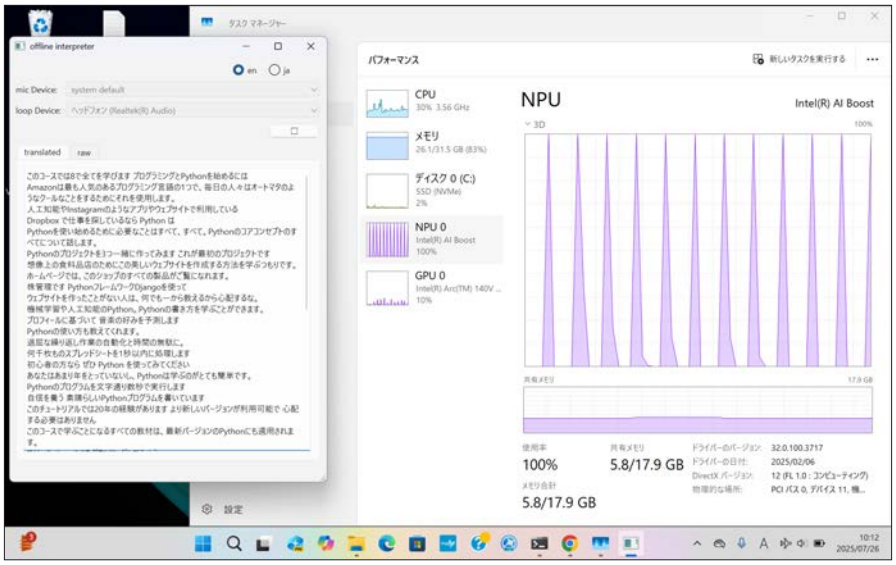
展望：AI PCアプリストア (AI Edge Hub)への展開

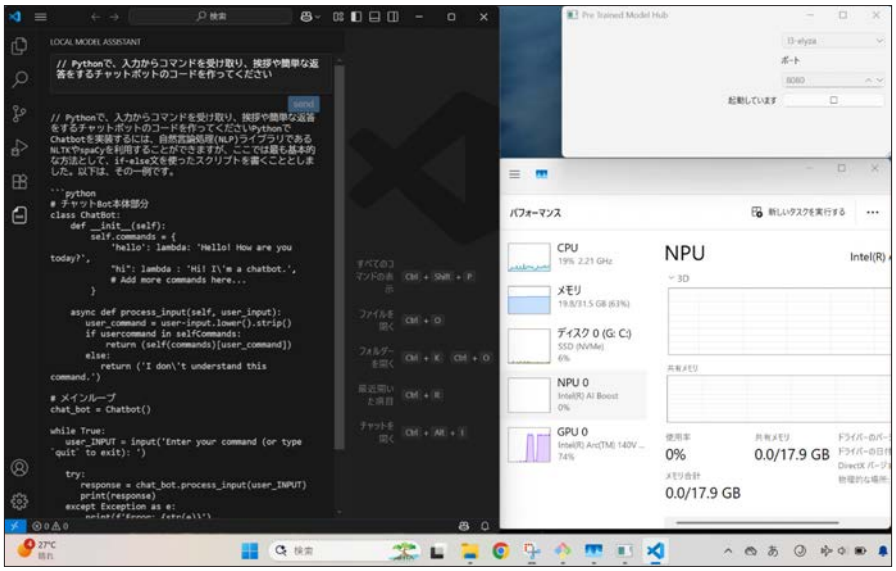
現在、K-kaleido (https://k-kaleido.com)ではこの仕組みを中核としたAI PC向けアプリストア(AI Edge Hub)の構築を進めており、2025年9月末には正式ローンチを予定しています。オフライン通訳、開発支援、業務ドキュメント生成など、ローカルAIの価値を最大限に活かす多彩なアプリケーションが順次展開される予定です。

ユーザーの手元で、セキュアかつ持続可能なAI体験を実現する

この思想に基づく「賢いアプリ」のエコシステムづくりこそが、K-kaleidoが目指すローカルAI時代の新たな挑戦です。







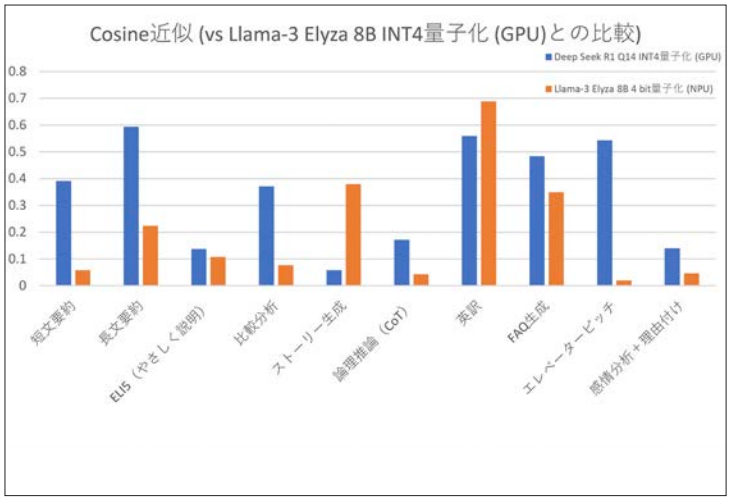
Appendix.1 データセンター由来の環境負荷と将来リスク

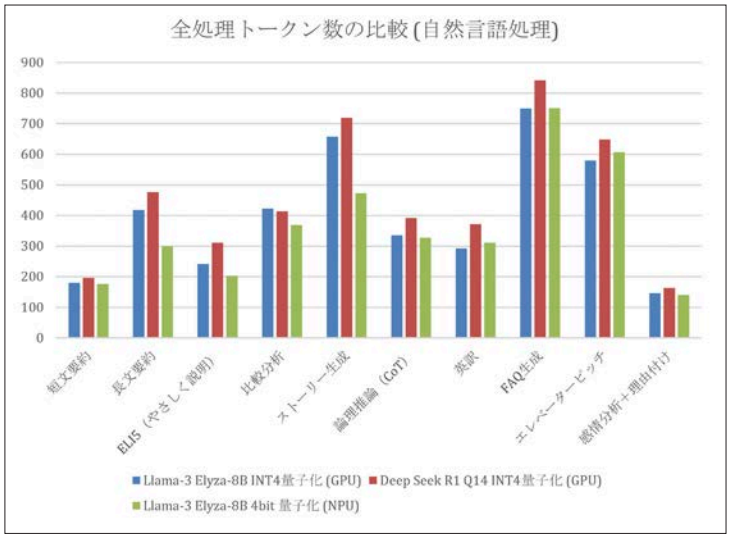
項目	現状と将来予測
電力消費 （データセンター全体）	約460TWh（2022年）→945TWh（2030年予測）※日本の総発電量（1000TWh規模）に近接
水使用量	米Google：2022年に51億リットル使用（冷却用途）
土地利用	米国最大手Hyperscalerではデータセンター 1拠点あたり数十万平方メートル
CO_2排出量	データセンター由来の排出：年間2億トン以上（航空業界に匹敵）
再エネ対応	企業によるRE100宣言増加も、需要増加スピードに対応しきれない地域あり
課題	電力逼迫リスク（AIワークロードによるGPU需要増）- インフラ整備の限界

Appendix.2 INT4量子化したSLMのプロンプト検証の詳細

A) 自然言語処理プロンプト

タスク名	タスク定義	評価観点
短文要約	約250文字（3文構成）の文章を、50〜80ワードで簡潔に要約する	要点抽出力、構成力、冗長表現の削除、焦点の明確さ
長文要約	約1100文字（3章構成）の業務報告文を、300ワード・5段落構成で再整理する	段落構造の維持、主要論点の網羅、再構成力、論理性
ELI5（かんたん説明）	技術的な概念（例：投機的実行）を、小学生にも伝わるようにやさしく言い換える	難解概念の咀嚼、具体例の提示、語彙の変換力
比較分析	2製品の仕様（性能・特長）を比較し、どちらを推奨すべきかを説明する	長所・短所の分類力、比較構造、推奨判断の妥当性
ストーリー生成	転生・戦争・兄弟対立などの設定に基づき、800ワード程度の物語を創作する	創造力、展開の自然さ、登場人物の心理描写の説得力
論理推論（Chain of Thought）	時刻に関する条件問題に対し、5ステップで推論し答えを導く	前提理解、数理的因果の構築、ステップの整合性
ビジネスメール英訳	約120文字の日本語メール文を、自然かつ丁寧なビジネス英語に訳す	意図の正確な翻訳、文体の自然さ、語彙と文法の適切性
FAQ自動生成	製品マニュアルから、ユーザーの疑問と回答を5件抽出・構成する	情報抽出力、QA構造化力、網羅性と実用性
エレベーターピッチ	社内提言資料を基に、経営層向け30秒（100トークン）プレゼン文を作成する	要点抽出力、訴求力、戦略的メッセージ性
感情分析＋理由付け	約100ワードのレビューを読み、感情（ポジ/ネガ）を判定し3つの理由を説明	感情分類の的確さ、根拠抽出、理由の一貫性と妥当性





シナリオ	参照文(Gold)の実務適合性	L3 Elyzaの実務適合性	コメント
短文要約	○	◎	L3 Elyzaは構文が明快で要点も正確。Goldと同等以上にそのまま実務転用可能。
長文要約	○	○	章構成の再整理が必要だが、要点網羅性はあり、実用には十分。
ELI5	△	△	両者とも具体例が抽象的で、小学生にとってのわかりやすさは今一歩。
比較分析	△	△	スペック列挙にとどまり、業務視点での推奨理由が弱く、再編集が必要。
ストーリー生成	△	△	物語構造が平板で感情表現も乏しいが、プロンプト改善で改善余地あり。
論理推論	◎	○	Goldは手順・根拠ともに明快。L3 Elyzaも答えに到達しており十分に使える。
ビジネスメール英訳	○	○	若干の表現硬さはあるが、文意伝達としては十分実務レベル。
FAQ生成	○	◎	網羅性と構成が明快で、Goldよりも実運用に近い構造となっている。
エレベーターピッチ	○	△	要点は含まれているが、説得力や圧縮構成にやや欠ける。リライト前提。
感情分析＋理由付け	△	○	分類は適切で構文も丁寧。Goldより説明が明快で、実用上は優れている。

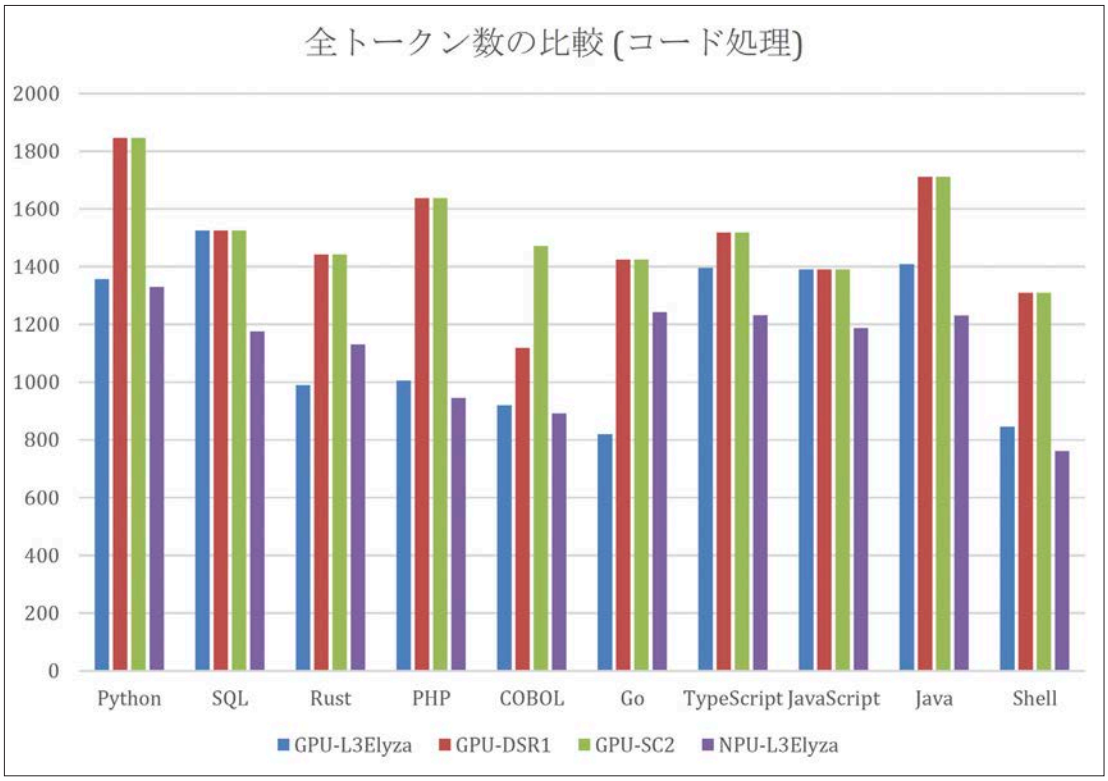
評価	解釈
◎	初期出力でも即実務転用可能。構造も意図も高水準。
○	一部編集で目的に到達。業務利用に支障なし。
△	プロンプト改善で改善可能。現状は素材レベル。
×	モデルの性質上、プロンプトを変えても目的達成は困難

タスク名	L3 Elyza NPU	DSR1 GPU
短文要約	△	○
長文要約	△	△
ELI5	△	△
比較分析	△	○
ストーリー生成	○	△
論理推論（Chain of Thoughts）	△	△
ビジネスメール英訳	○	○
FAQ自動生成	△	◎
エレベーターピッチ	△	○
感情分析＋理由付け	○	○

記号	意味
◎	出力がL3 Elyza GPUと非常に近く、構成・語彙・情報量のバランスも良好
○	意図は近いが、表現の粒度や構成にやや違いがある
△	出力の焦点や構成がズレており、再調整・補足が必要な可能性がある

B) プログラミングコード処理プロンプト

タスク名	タスク定義	評価観点
Pythonでのデータ整理処理	Pythonでデータを収集・変換・保存するETL処理を記述します。構造が明確で、再利用可能な設計がなされているかを確認します。	データ処理ロジックの正確性、処理順の理解、記述の簡潔さ
Go言語によるAPI実装	GoでRESTfulなAPIを実装し、リクエスト処理・ルーティング・エラーハンドリングを構成します。	型の扱い、ルーティングの構造化、REST設計の理解
Javaによるストリーム処理	Java Stream APIを活用し、リストなどの大量データを絞り込み・並べ替え・集計する処理を構築します。	関数型構文（ラムダ、map、filterなど）の使いこなし、処理の簡潔さ
JavaScriptによるUI動作設計	JavaScriptでボタンなどのUI操作に応じた非同期通信とDOM更新処理を設計します。通信エラー時のフォールバック動作も含まれます。	ユーザー操作への反応処理、非同期制御、DOM更新の正確さ
TypeScriptによる依存性注入構成	TypeScriptを用いて、依存性注入（DI）に基づく機能分離と柔軟な構成を設計します。	モジュール設計力、柔軟性、保守性に優れた構造の理解
Bashスクリプトによるデータ処理パイプライン	Bashで複数のコマンドを組み合わせ、テキストファイルの整形・変換・集計を自動化します。	コマンド構成力、パイプ・条件分岐の理解、作業自動化の正確性
PHPによるMVC構造の実装	PHPでモデル（データ）・ビュー（画面）・コントローラ（制御）を明確に分けたWebアプリ構造を設計します。	アーキテクチャの理解、役割分離、スパゲッティ化の回避
Rustによるデータ処理ロジックの設計	Rustで所有権と安全性に配慮しつつ、高速なデータ処理ロジックを設計します。	メモリ安全性、型システム活用、関数・構造体の設計力
COBOLによるバッチジョブ定義	COBOLでファイルを読み込み、条件に基づいてレコードを分類・出力するバッチ処理を記述します。	古典的業務ロジックの再現力、手続き的構造の正確な理解
SQLによる高度なデータ検索クエリ生成	SQLで複数テーブルをJOINし、集計・サブクエリを組み合わせた複雑な条件検索を行うクエリを記述します。	データベース構造の理解、正確な論理表現、実行効率への配慮



シナリオ	Elyza (GPU) vs SC2 (GPU)	Elyza (GPU) vs Elyza (NPU)
Python_ETL	L3 Elyza は各処理ブロックの役割が明確に分けられており、データ変換の流れが追いやすい構成だった。SC2は機能の重複や冗長な記述が目立ち、全体の構造がつかみにくかった。 実用上はElyzaの出力の方が完成度・保守性ともに高く、SC2は網羅的だが構造が不明瞭で、修正・保守を前提とする現場ではそのまま使うのは難しい	NPUはログ出力の位置や処理順序について補足があり、設計のヒントとなる内容が含まれていた。GPUは具体的な処理記述が中心だったが、背景説明は少なめ。ただしNPUは出力量が少なく、コードそのものは出力されず、補助的な用途にとどまる。 設計フェーズではNPUの示唆が役立つが、実装にはGPUの出力をベースにするのが現実的。
SQL_JOIN	L3 Elyza はJOIN対象テーブルの関係性を明確に説明し、WHERE句の使い方やインデックス設計にも触れていた。SC2は複雑なクエリが一塊で出力され、修正や保守にはやや不向きな印象だった。	NPUでは結合条件の分解やサブクエリの整理について補足があり、設計方針の再考に役立った。GPU出力は実装としては正確だったが、意図の説明は乏しかった。 実用上はGPU出力の活用が現実的。
Rust_ownership	L3 Elyza は変数スコープとライフタイムの整理がなされており、初心者でも追いやすい構成だった。SC2は所有権と参照が混在し、動作はしても理解には時間を要する内容だった。	NPUは変数の使い回しや参照の分離について補足があり、設計改善のヒントになった。GPUはすぐ動く形だったが、理由説明が少なく、NPUと組み合わせて理解が深まる。
PHP_MVC	L3 Elyza はViewとControllerの責務を分離し、処理と表示が整理されていた。SC2は認証やフォーム処理が詰め込まれていて、全体の流れが把握しにくかった。 実用上はElyzaの出力の方が完成度・保守性ともに高く、SC2は参考用途にとどまる。	NPUはセッション管理やトークン処理の分離について補足があり、設計方針の改善につながる示唆が含まれていた。ただしコード出力はなかった。
COBOL_multi	L3 Elyza は段階的な処理構成で、帳票出力までの流れが把握しやすかった。SC2は手続きが詰まっており、読解や修正には労力がかかる構成だった。	NPUはデータ定義と処理分離の観点でコメントがあり、保守性向上の観点で参考になった。GPUは従来通りの記述で即時実行には向くが、 説明は簡素だった。
Shell_jq	L3 Elyza は必要なフィールドだけを抽出しており、シンプルかつ拡張しやすい構成だった。SC2は長大で複雑なパイプ構文が続き、可読性や再利用性に難があった。	NPUではフィルタ順序の整理や構文簡素化の提案があり、スクリプト設計の学習に適した内容だった。ただし出力全体は短く、具体コードの出力はない。 実用上はGPU出力の活用が現実的。
Go_HTTP	L3 Elyza はエラーハンドリングやルーティングを関数分離しており、保守性が高い構成だった。SC2はロジックが1つのハンドラーに集中しており、テストや拡張がしづらい印象だった。	NPUではパラメータ検証や分岐設計の工夫について補足があり、全体設計の見通しを良くする内容だった。GPUは動作として問題はないが、 意図の説明は控えめだった。 実用上はGPU出力の活用が現実的。
JS_UI	L3 Elyza はfetch処理とDOM更新の責務分離が意識されており、操作の流れが明確だった。SC2も似た構成だが、関数の再利用性やセキュリティ配慮が薄く、改善案がやや抽象的だった。 実用面での優劣は明確にしづらい。	NPU出力ではURLパラメータ処理に対するセキュリティ上の工夫が補足されていた。GPUは処理構造を重視し、実用的な改善案が提示されていたが、NPUはより設計上の注意に絞った内容だった。ただし出力量は少なく、コード出力は含まれていない。
Java_Stream	L3 Elyza はラムダ構文とストリーム操作の構成が整理されており、読みやすく保守しやすい印象だった。SC2は一連の操作が連続して記述されており、流れは分かるがSC2はすべての操作が1行に詰め込まれており、後から条件変更やフィルタ挿入を行う際に分解が必要となる。編集や保守のしやすさの観点では課題が残る。	NPUはフィルタ条件や中間操作の分離について設計的な補足があり、GPU出力の実装と補完関係にある。ただし出力量は少なめ。
TS_DI	L3 Elyzaは依存関係の注入構造が明確で、再利用性とテスト容易性が確保されていた。SC2は依存がベタ書きされており、拡張やモック導入には工夫が必要だった。	NPUは設計上の依存性の扱い方に関する補足があり、GPU出力の補完として有効だった。NPUは出力量が少なく、コードそのものは出力されていないが、観点としての価値は高い。設計フェーズではNPUの示唆が役立つが、実装にはGPUの出力をベースにするのが現実的。