# バイブコーディングとAI統合エコシステムの最適化：2026年版包括的戦略レポート

## 1. 序論：バイブコーディングがもたらすソフトウェア開発のパラダイムシフト

### 1.1 「バイブコーディング」の概念的定義と歴史的背景

「バイブコーディング（Vibe Coding）」という用語は、元OpenAIの研究者であり、Teslaのオートパイロット部門の責任者を務めたアンドレイ・カルパシー（Andrej Karpathy）によって広められた概念であり、2025年から2026年にかけてソフトウェア開発の現場における支配的なトレンドとして定着した 1。伝統的なプログラミングが、構文（Syntax）の正確な記述と論理構造の厳密な構築を人間に要求したのに対し、バイブコーディングは「自然言語による意図（Intent）の記述」と「AIによる実装の委任」を核とする。

このパラダイムにおいて、人間の役割は「コードを書くこと（Writing）」から「AIの出力を管理・レビューすること（Managing & Reviewing）」へと移行する。開発者は、詳細なアルゴリズムの実装よりも、システム全体のアーキテクチャ、ユーザー体験（UX）、およびビジネスロジックの整合性に注力することが求められる。カルパシーは、これを「プログラミング言語としての英語（English as the hottest new programming language）」と表現しており、自然言語処理能力が飛躍的に向上した大規模言語モデル（LLM）の台頭によって初めて可能となった 2。

### 1.2 現代の開発環境における「課金枠」の戦略的価値

本レポートの読者は、Claude Pro、ChatGPT Plus、Google One AI Premium、Z.AI LITEプランという、現時点で世界最高峰のAIサービスに対する有料サブスクリプションを保有している。これらは単なる「チャットボットの利用権」ではなく、それぞれが異なる強みを持つ「計算資源（Compute Resources）」および「推論エンジン（Reasoning Engines）」のポートフォリオである。

* **Z.AI LITE**: コストパフォーマンスに優れた「高馬力・量産型」コーディングエンジン（GLM-4.7）。
* **Claude Pro**: 繊細な文脈理解と最高峰のコード品質を提供する「天才肌」のエンジニア（Claude 3.5 Sonnet）。
* **Google One AI Premium**: 圧倒的なコンテキストウィンドウ（200万トークン）を持つ「巨大記憶」アーカイブ（Gemini 1.5 Pro）。
* **ChatGPT Plus**: 深い推論能力と多角的な支援機能を持つ「論理的」アドバイザー（o1 / GPT-4o）。

これらの資産を個別に運用するのではなく、相互に連携させた「統合エコシステム」として運用することで、単一のサービスでは到達不可能な生産性と品質を実現することが、本レポートの主眼である。

## 2. 保有サブスクリプション資産の徹底分析と最適化

### 2.1 Z.AI LITEプラン：隠れた最強のコーディングエンジン

多くのユーザーがZ.AI（Zhipu AI）を単なる「安価な代替手段」と見なしているが、2025年後半から2026年にかけてのアップデートにより、その評価は劇的に変化した。特に「GLM-4.7」モデルの投入と、コーディングツール（Cursor, Cline等）への最適化プランは、バイブコーディングの実践において中核を担うポテンシャルを持つ 3。

#### 2.1.1 GLM-4.7の技術的特性とベンチマーク評価

Z.AIのLITEプランで利用可能な最新モデル「GLM-4.7」は、コーディングおよび推論タスクにおいて、Claude 3.5 SonnetやGPT-4oと直接競合する性能を有している。

| **ベンチマーク指標** | **GLM-4.7** | **Claude 3.5 Sonnet** | **GPT-4o** | **備考** |
| --- | --- | --- | --- | --- |
| **SWE-bench Verified** | **73.8%** | 49.0% (旧Ver) / 高水準 | 高水準 | 実際のGitHubの課題解決能力を測定。GLM-4.7は特にエージェント的な自律解決能力が高い 5。 |
| **HumanEval** | **90%+** | 92.0% | 90.2% | 関数レベルのコード生成能力は全モデルが飽和状態だが、GLM-4.7もトップティアに位置する。 |
| **MMLU-Pro** | **84.3%** | 77.6% | - | 複雑な推論タスクにおけるスコア。論理的整合性の高さを示す 5。 |

特筆すべきは、GLM-4.7が「Thinking（思考）」プロセスを強化しており、複雑な指示に対しても内部で推論ステップを踏んでから回答を出力する点である 6。これにより、単なるコード補完を超えた、アーキテクチャレベルの提案が可能となっている。

#### 2.1.2 コスト構造と利用制限の戦略的優位性

Z.AI LITEプランの最大の魅力は、その圧倒的なコストパフォーマンスと利用枠の大きさにある。

* **価格**: キャンペーン適用時で月額3ドル〜（四半期/年払い）。通常価格でも非常に安価 3。
* **利用制限**: **5時間ごとに約120プロンプト** 4。
  + **比較**: Claude Proの制限（詳細非公開だが、概ね5時間で数十メッセージ程度、特に長文コンテキストでは激減する）と比較して、約3倍の容量を持つと謳われている 4。
  + **実質的な意味**: バイブコーディングでは、AIにコードを書かせ、エラーが出たら修正させ、さらにテストを書かせるといった「試行錯誤のループ」が頻発する。Claude Proではこのループを数回回すだけで制限に達してしまうが、Z.AI LITEなら、その3倍以上の試行が可能である。これは「質より量」が必要な初期実装フェーズにおいて決定的な差となる。

#### 2.1.3 API統合の柔軟性

Z.AIはOpenAI互換のAPIエンドポイントを提供しており、Cursor、Cline、Roo Codeなどの主要なAIコーディングツールにシームレスに統合できる 9。ユーザーはZ.AIの専用アプリを使う必要はなく、使い慣れたIDEの裏側でGLM-4.7を動かすことができる。これは、ユーザー体験（UX）を変えずに計算資源だけを増強できることを意味する。

### 2.2 Claude Pro：品質とMCPの司令塔

Anthropic社のClaude 3.5 Sonnetは、コードの「品質」「安全性」「保守性」において依然として信頼性が高い。Proプランの価値は、単なるチャット利用を超え、**Projects機能**と**Artifacts機能**、そしてローカルアプリとしての**Claude Desktop**にある。

#### 2.2.1 Artifactsによる「即時プレビュー」革命

Artifacts機能は、生成されたコード（Reactコンポーネント、HTML/CSS、SVGグラフィックスなど）を、チャットウィンドウの横で即座にレンダリングする機能である 11。

* **バイブコーディングでの役割**: 自然言語で「青を基調としたダッシュボードを作って」と指示し、即座に画面を確認。気に入らなければ「もっと丸みを持たせて」と指示して修正。このサイクルは、従来のエディタでファイルを作成し、ブラウザをリロードする手間を完全に排除する。フロントエンド開発における「Vibe（雰囲気）」の調整には不可欠な機能である。

#### 2.2.2 Projects機能によるコンテキスト管理

Claude ProのProjects機能では、関連するコードファイルやドキュメントを「知識ベース（Knowledge Base）」として登録できる。

* **戦略的利用**: プロジェクト固有の「コーディング規約」「デザインシステム」「API仕様書」をProjectに登録しておくことで、Claudeはその制約を常に意識したコードを出力するようになる。これは、大規模プロジェクトにおけるコードの一貫性を保つ上で重要である。

#### 2.2.3 Claude DesktopとMCPの統合

Claude Desktopアプリは、**Model Context Protocol (MCP)** のホストとして機能する 13。これにより、ClaudeはローカルファイルシステムやGitリポジトリ、社内データベースに直接アクセス可能となる。Webブラウザ上のClaudeとは異なり、ローカル環境の「オペレーティングシステム」として振る舞うことができるため、ファイル操作や複雑な自動化タスクにおいて真価を発揮する。

### 2.3 Google One AI Premium：200万トークンの「巨大な器」

Gemini 1.5 Pro（Gemini Advanced）の最大の特徴は、他社を圧倒する\*\*200万トークン（2M Context Window）\*\*である 15。

#### 2.3.1 「全読み」が変える開発スタイル

通常のLLM（128k〜200kトークン）では、大規模なリポジトリを扱う際にRAG（検索拡張生成）やファイルの選別が必要となる。しかし、Gemini 1.5 Proでは、数万行〜数十万行のコードベース全体を一度にコンテキストに乗せることが可能である。

* **大規模リファクタリング**: 「プロジェクト全体の変数命名規則をCamelCaseからSnakeCaseに変更し、影響範囲を全てリストアップして」といった、全体を把握していなければ不可能な指示が通る。
* **未知のコード解析**: オープンソースライブラリのソースコードを丸ごと読ませ、「このライブラリの認証フローがどうなっているか図解して」と指示することで、ドキュメントが不十分なライブラリの解析時間を大幅に短縮できる。
* **マルチモーダル入力**: 画面遷移図（画像）や操作動画（動画）をアップロードし、「このUIデザインに基づいたフロントエンドコードを生成して」と指示できる 17。

#### 2.3.2 Googleエコシステムとの連携

Google Workspace（Docs, Drive, Gmail）との連携も強力であり、仕様書がGoogle Docsにある場合、それを直接参照してコードを生成させることができる 15。

### 2.4 ChatGPT Plus：推論の「o1」と多機能ハブ

OpenAIのChatGPT Plusは、汎用性と「最後の砦」としての推論能力に優れる。

#### 2.4.1 o1 (OpenAI o1) の「思考力」

「Thinking Model」であるo1は、即答型のモデルが苦手とする、複雑な論理パズルやアルゴリズムの最適化、原因不明のバグ特定において無類の強さを発揮する 19。

* **デバッグの最終手段**: ClaudeやGeminiでも解決できない競合状態（Race Condition）やメモリリークの問題に対し、o1に「じっくり考えて（Think for a while）」と指示することで、人間でも気づかない論理的欠陥を指摘してくる場合がある。

#### 2.4.2 Canvas機能

ChatGPT Canvasは、ClaudeのArtifactsに似ているが、より「ドキュメント編集」や「コードレビュー」に特化している 11。

* **インライン編集**: 生成されたコードの特定行を選択し、「ここをもう少し効率化して」とコメントを入れるような感覚で修正指示が出せる。ドキュメント作成やブログ執筆にも適しており、開発に伴う付帯作業を効率化する。

#### 2.4.3 Voice Mode (Advanced Voice)

バイブコーディングの究極形として「音声によるコーディング」がある。散歩中や移動中に、ChatGPTの音声モードに対して「今作っているアプリのアイデア」を話し、それを要件定義としてまとめさせる。この「ハンズフリー・バイブコーディング」は、クリエイティブな発想を妨げない強力な手法である。

## 3. バイブコーディングのための「統合ツール環境（The Stack）」

有料サブスクリプション（モデル）を最大限に活かすためには、それらを統合する「インターフェース（IDE）」の選定と設定が不可欠である。

### 3.1 Cursor：現行の王者とZ.AI統合

Cursorは、VS CodeをフォークしてAI機能をネイティブ統合したIDEであり、バイブコーディングの実践者にとって標準的なツールとなっている 21。

#### 3.1.1 Composer機能の活用

Cursorの「Composer（Ctrl+I / Cmd+I）」は、複数ファイルにまたがる変更を自然言語で指示し、一括で適用できる機能である。これこそがバイブコーディングの真骨頂である。

* **Z.AI GLM-4.7の設定**: Cursorの設定画面で「OpenAI Compatible」を選択し、Base URLに https://api.z.ai/api/coding/paas/v4 を入力、APIキーを設定する 9。モデル名として GLM-4.7 を追加する。
* **戦略**: Composerのデフォルトモデルを「GLM-4.7」に設定する。これにより、日常的な機能追加や修正はZ.AIの豊富な枠を使って行い、Claude Proの制限を温存できる。GLM-4.7は120プロンプト/5時間という余裕があるため、Composerで「とりあえず試す」という贅沢な使い方が可能になる。

#### 3.1.2.cursorrules による制御

プロジェクトルートに .cursorrules ファイルを配置することで、AIの振る舞いを制御できる。

* **内容**: 「常にTypeScriptを使用すること」「コメントは日本語で書くこと」「テストフレームワークはJestを使うこと」などを記述しておくと、GLM-4.7であれClaudeであれ、そのルールに従ったコードを出力する。これにより、モデル間の出力のブレを最小限に抑えられる。

### 3.2 Windsurf：次世代のAgentic IDE

Windsurfは、Codeium社が開発した新しいIDEであり、Cursorの強力な競合である。「Cascade」と呼ばれるフロー機能が特徴で、ユーザーの行動を先読みするような体験を提供する 21。

#### 3.2.1 Deep Context Awareness

Windsurfは、プロジェクト全体のインデックス化（理解）能力においてCursorを凌ぐ場合がある。変数の定義元や参照先を深く理解し、ユーザーが明示的にファイルを開いていなくても、関連コードを修正候補として提示する。

* **BYOK (Bring Your Own Key)**: Windsurfは最近のアップデートで、無料プランでもAPIキーの持ち込み（BYOK）が可能になった 23。Z.AIのAPIがOpenAI互換であることを利用し、WindsurfでもGLM-4.7を利用できる可能性がある（要検証だが、OpenAI互換エンドポイントの設定が可能であれば理論上動作する）。

### 3.3 Bolt.new & Bolt.diy：Webブラウザでのフルスタック生成

StackBlitzが提供するBolt.newは、環境構築不要でブラウザ上でReact/Node.jsアプリを生成・実行できるツールである。

#### 3.3.1 Bolt.diy による「制限突破」

公式のBolt.newには無料枠の制限があるが、オープンソース版の **Bolt.diy** を使用すれば、自分のAPIキーを使って無制限に利用できる 24。

* **ローカル構築**: DockerまたはNode.js環境でBolt.diyをローカル起動する。
* **Z.AI / Gemini APIの統合**: Bolt.diyの設定で、Z.AIのAPIキーやGeminiの無料APIキーを設定する。これにより、ブラウザベースの高速なプロトタイピング環境が、実質無料で（またはZ.AIの定額枠内で）手に入る。
* **用途**: 「新しいアプリのアイデアを思いついたが、環境構築が面倒」という場面で、Bolt.diyを立ち上げ、自然言語だけでプロトタイプを完成させる。その後、生成されたコードをダウンロードし、Cursorで本格的な開発に移行するフローが最強である。

### 3.4 Cline (旧 Claude Dev) & Roo Code

VS Codeの拡張機能として動作するオープンソースのエージェントツール。

* **透明性とコスト管理**: 使用したトークン数とコストがリアルタイムで表示されるため、Z.AI LITEプランの消費状況を正確に把握できる。
* **マルチモデル運用**: 1つのタスク内で、計画立案にはClaude 3.5 Sonnet、実装にはGLM-4.7、レビューにはDeepSeek R1といったように、モデルを細かく切り替えて運用するのに適している。

## 4. ローカルLLMと無料API/MCPの「隙間埋め」戦略

有料プランだけではカバーしきれない部分（プライバシー、オフライン、超高速レスポンス、特殊ツール連携）を、無料およびローカルのリソースで補完する。

### 4.1 DeepSeek R1 (Local)：ローカルで動く「叡智」

2026年現在、DeepSeek R1はその高い推論能力とオープンな重み公開により、ローカルLLMの決定版となっている 26。

#### 4.1.1 ハードウェア要件とモデル選択

ユーザーの環境に合わせて最適なモデルサイズを選択する。

| **モデル名** | **パラメータ数** | **必要VRAM/メモリ (4-bit量子化)** | **推奨ハードウェア** | **用途** |
| --- | --- | --- | --- | --- |
| **DeepSeek-R1-Distill-Qwen-1.5B** | 1.5B | ~2GB | 全てのPC、スマホ | 超高速な補完、簡単なロジック |
| **DeepSeek-R1-Distill-Llama-8B** | 8B | ~6GB | MacBook Air (8GB/16GB), RTX 3060 | 一般的なコーディング支援、チャット |
| **DeepSeek-R1-Distill-Qwen-32B** | 32B | ~20GB | MacBook Pro (32GB/36GB), RTX 3090/4090 | **推奨ライン**。高度な推論とコード生成 |
| **DeepSeek-R1-Distill-Llama-70B** | 70B | ~40GB | Mac Studio (64GB), Dual RTX 3090/4090 | GPT-4級の性能をローカルで実現 |

* **導入**: **Ollama** を使用するのが最も手軽である 28。ollama run deepseek-r1:32b コマンドで起動し、CursorやClineから localhost:11434 を指定して接続する。
* **バイブコーディングでの価値**: インターネットに接続できない環境や、絶対に社外に出せない秘匿性の高いコードを扱う際、DeepSeek R1は最強のパートナーとなる。また、API課金を気にせず「無限に思考させる（Chain of Thought）」ことができるため、難問の解決にも適している。

### 4.2 無料API枠の徹底活用

有料プランの枠を節約するために、各社が提供する無料API枠（Free Tier）を賢く利用する。

* **Google AI Studio (Gemini API)**:
  + **Gemini 1.5 Flash**: 高速かつ安価（一定枠まで無料）。Bolt.diyでの初期生成や、簡単なスクリプト生成に最適 29。
  + **Gemini 1.5 Pro**: 無料枠でも利用可能（レートリミットあり）。2MコンテキストをAPI経由で試せる貴重な環境。
* **Groq**:
  + LPU（Language Processing Unit）による超高速推論。Llama 3.3 70Bなどを爆速で動かせる 30。チャットボットの応答速度テストや、リアルタイム性が求められるツールのバックエンドとして無料枠を活用する。
* **OpenRouter Free Models**:
  + DeepSeek R1の無料提供版や、その他の実験的モデルにアクセスできる 31。Clineの設定でOpenRouterを選択し、「Free」タグのついたモデルを指定することで、完全無料でAIコーディングが可能になる。

### 4.3 Model Context Protocol (MCP) の無料ツール群

Claude DesktopやClineで利用可能なMCPサーバーは、開発環境を「AIの身体」へと拡張する。これらは基本的にオープンソースであり、無料で利用可能である。

#### 4.3.1 必須導入MCPサーバー

1. **Filesystem Server (@modelcontextprotocol/server-filesystem)**:
   * **機能**: ローカルの指定フォルダ内のファイルを読み書きする権限をAIに与える。
   * **用途**: 「このフォルダ内の画像ファイルを全部リサイズするスクリプトを書いて実行して」といった指示が可能になる 14。
2. **Git Server**:
   * **機能**: Gitリポジトリの履歴閲覧、差分確認、コミット作成。
   * **用途**: 「前回のコミットでエンバグしたっぽいから、差分を見て原因を特定して」という指示が可能になる。
3. **Brave Search Server**:
   * **機能**: Web検索を行い、最新情報を取得する。
   * **用途**: 「Reactの最新バージョンのドキュメントを検索して、この非推奨メソッドの代替案を教えて」と指示する。
4. **PostgreSQL / SQLite Server**:
   * **機能**: データベースへのクエリ実行とスキーマ確認。
   * **用途**: 「ユーザーテーブルの構造を確認して、新しいカラムを追加するマイグレーションファイルを書いて」と指示する。
5. **Fetch Server**:
   * **機能**: 指定したURLのWebページコンテンツをテキストとして取得する 13。
   * **用途**: 「この技術ブログの内容を要約して、実装の参考にして」と指示する。

## 5. 究極の統合最適化戦略：シナリオ別ワークフロー

ここまで分析した資産を組み合わせた、具体的な運用シナリオを提言する。

### シナリオA：新規Webサービスの爆速立ち上げ（Zero to One）

**目的**: アイデアから動くプロトタイプを最短で作る。

1. **構想 (Voice)**: **ChatGPT Plus (Voice Mode)** で散歩しながらアイデアを壁打ちし、要件定義のメモを作成させる。
2. **骨格生成 (Web)**: **Bolt.diy (Local)** を起動し、**Z.AI (GLM-4.7)** または **Gemini 1.5 Flash (Free)** を設定。要件定義を貼り付け、アプリの全体像を生成・実行させる。UIをプレビューしながら自然言語で修正。
3. **本格実装 (IDE)**: 生成されたコードをダウンロードし、**Cursor** で開く。
4. **開発 (Coding)**: **Z.AI (GLM-4.7)** をComposerに設定し、機能を作り込む。
5. **仕上げ (UI)**: **Claude 3.5 Sonnet (Pro)** に切り替え、Artifacts的感覚でUIの微調整や複雑なロジックの仕上げを行う。

### シナリオB：大規模レガシーコードのリファクタリング

**目的**: 誰も触りたくない巨大なスパゲッティコードを安全に改修する。

1. **理解 (Analysis)**: リポジトリ全体を **Google AI Studio (Gemini 1.5 Pro)** にアップロード。「このプロジェクトのアーキテクチャ図とデータフロー図を描いて」と指示し、全体像を把握する。
2. **計画 (Plan)**: **Claude 3.5 Sonnet (Pro)** に、Geminiの分析結果と対象コードを渡し、リファクタリング計画を立案させる。
3. **実行 (Refactor)**: **Cursor + Z.AI (GLM-4.7)** で、ファイル単位の書き換えを実行。.cursorrules に厳密なルールを設定しておく。
4. **検証 (Review)**: 変更後のコードを **DeepSeek R1 (Local)** に読ませ、「論理的なバグや退行（Regression）がないか」をローカルでじっくり推論させる。

### シナリオC：APIコストゼロ・完全オフライン開発

**目的**: ネットがない環境や、コストをかけられない状況での学習・実験。

1. **環境**: **VS Code + Cline** + **Ollama (DeepSeek R1 32B)**。
2. **実行**: 全てのリクエストをローカルのDeepSeek R1に投げる。速度はマシンスペックに依存するが、回答の質はGPT-4級。コストは電気代のみ。

## 6. 結論と2026年への展望

本レポートの調査により、ユーザーが保有する「Z.AI LITE」「Claude Pro」「Google One AI Premium」「ChatGPT Plus」は、現在のAI開発環境において考えうる\*\*最強の組み合わせ（ロイヤルストレートフラッシュ）\*\*であることが確認された。これらに不足している要素はない。唯一の課題は「使い分け」である。

**【提言まとめ】**

1. **メインエンジンをZ.AI (GLM-4.7) に切り替えよ**: 日常的なCursorでのコーディングは、Claude Proのクレジットを消費せず、Z.AIの豊富な枠（120回/5時間）で行うべきである。設定は今すぐ行うこと。
2. **ClaudeとGeminiは「スペシャリスト」として温存せよ**: ClaudeはUIと最終品質のため、Geminiは大量のコンテキスト処理のために使う。これらを雑用（単純な関数作成など）に使ってはならない。
3. **ローカルとBolt.diyで「実験」を加速せよ**: 失敗しても痛くない環境（DeepSeek R1, Bolt.diy）を持つことで、バイブコーディングの試行回数を飛躍的に増やせる。

この戦略を実行することで、ユーザーは追加のコストを支払うことなく、開発スピードと品質を極限まで高めることができるだろう。2026年は「AIに書かせる」時代から「AIを指揮して創り上げる」時代への転換点であり、本レポートの戦略はその最前線に立つための羅針盤となるものである。

# 付録資料：技術的詳細設定ガイド

## A. Cursor × Z.AI 接続設定

Z.AIのAPIをCursorで利用するための正確な手順（2025/2026年版）。

1. **APIキー取得**: [Z.AI Open Platform](https://open.bigmodel.cn/) にログインし、APIキーを作成する。
2. **Cursor設定**:
   * 画面右上の歯車アイコン > Models タブを開く。
   * OpenAI API Key の欄に取得したZ.AIのキーを入力（すでに入力済みの場合はOverrideを使用）。
   * **Override OpenAI Base URL**: ここが最重要。https://api.z.ai/api/coding/paas/v4 と入力する。通常のチャット用エンドポイントとは異なる場合があるため、必ずCoding用エンドポイントを確認すること 9。
   * **Model Name**: Add model をクリックし、GLM-4.7 を追加。念のため glm-4.7（小文字）も追加し、トグルをオンにする。
   * **Verify**: 設定後、チャット欄で @GLM-4.7 を選択し、簡単な挨拶をして動作確認を行う。

## B. MCP サーバー設定 (claude\_desktop\_config.json)

macOSの場合の標準的な設定ファイルのパスは ~/Library/Application Support/Claude/claude\_desktop\_config.json である。Windowsの場合は %APPDATA%\Claude\claude\_desktop\_config.json。

JSON

{  
 "mcpServers": {  
 "filesystem": {  
 "command": "npx",  
 "args":  
 },  
 "git": {  
 "command": "npx",  
 "args": [  
 "-y",  
 "@modelcontextprotocol/server-git"  
 ]  
 },  
 "fetch": {  
 "command": "npx",  
 "args": [  
 "-y",  
 "@modelcontextprotocol/server-fetch"  
 ]  
 }  
 }  
}

※ yourusername は自身のユーザー名に置き換えること。また、node と npm がインストールされていることが前提である。

## C. ローカルDeepSeek R1用ハードウェア・スペック表

GPUメモリ（VRAM）が不足する場合、メインメモリ（RAM）で代用できるが、速度は大幅に低下する（MacのUnified Memoryは例外で、比較的高速に動作する）。

| **モデル** | **推奨GPU (NVIDIA)** | **推奨Mac (Apple Silicon)** | **メモリ目安** |
| --- | --- | --- | --- |
| **DeepSeek-R1-Distill-Qwen-1.5B** | GTX 1650 以上 | M1 以上 | 4GB |
| **DeepSeek-R1-Distill-Llama-8B** | RTX 3060 (12GB) | M1/M2/M3 (8GB/16GB) | 8GB - 16GB |
| **DeepSeek-R1-Distill-Qwen-14B** | RTX 3080/4070 (12GB+) | M1/M2/M3 Pro (16GB/32GB) | 16GB - 24GB |
| **DeepSeek-R1-Distill-Qwen-32B** | RTX 3090/4090 (24GB) | M1/M2/M3 Max (32GB/64GB) | 32GB - 48GB |
| **DeepSeek-R1-Distill-Llama-70B** | A6000 / Dual 3090 | M1/M2 Ultra (64GB/128GB) | 64GB+ |

**(レポート終了)**

#### 引用文献

1. Vibe Coding Explained: Tools and Guides | Google Cloud, 1月 9, 2026にアクセス、 <https://cloud.google.com/discover/what-is-vibe-coding>
2. Vibe coding - Wikipedia, 1月 9, 2026にアクセス、 <https://en.wikipedia.org/wiki/Vibe_coding>
3. GLM Coding Plan — AI Coding Powered by GLM-4.7 for Agents ..., 1月 9, 2026にアクセス、 <https://z.ai/subscribe>
4. Overview - Z.AI DEVELOPER DOCUMENT, 1月 9, 2026にアクセス、 <https://docs.z.ai/devpack/overview>
5. Claude 3.5 Sonnet vs GLM-4.7 - LLM Stats, 1月 9, 2026にアクセス、 <https://llm-stats.com/models/compare/claude-3-5-sonnet-20241022-vs-glm-4.7>
6. GLM-4.7: Advancing the Coding Capability - Z.ai Chat, 1月 9, 2026にアクセス、 <https://z.ai/blog/glm-4.7>
7. Z.ai Coding Plan, 1月 9, 2026にアクセス、 <https://z.ai/landing-page/coding-plan>
8. FAQs - Overview - Z.AI DEVELOPER DOCUMENT, 1月 9, 2026にアクセス、 <https://docs.z.ai/devpack/faq>
9. Other Tools - Overview - Z.AI DEVELOPER DOCUMENT, 1月 9, 2026にアクセス、 <https://docs.z.ai/devpack/tool/others>
10. Claude Code - Overview - Z.AI DEVELOPER DOCUMENT, 1月 9, 2026にアクセス、 <https://docs.z.ai/scenario-example/develop-tools/claude>
11. Claude vs ChatGPT for Coding: Which AI Wins in 2025? - Leanware, 1月 9, 2026にアクセス、 <https://www.leanware.co/insights/claude-vs-chatgpt-coding>
12. ChatGPT Canvas vs Claude Artifacts for Programming: A Comprehensive Comparison, 1月 9, 2026にアクセス、 <https://algocademy.com/blog/chatgpt-canvas-vs-claude-artifacts-for-programming-a-comprehensive-comparison/>
13. The MCP Server Stack: 10 Open-Source Essentials for 2026 | by TechLatest.Net | Dec, 2025, 1月 9, 2026にアクセス、 <https://medium.com/@techlatest.net/the-mcp-server-stack-10-open-source-essentials-for-2026-cb13f080ca5c>
14. Connect to local MCP servers - Model Context Protocol, 1月 9, 2026にアクセス、 <https://modelcontextprotocol.io/docs/develop/connect-local-servers>
15. ChatGPT vs Gemini vs Claude: Best AI Model in 2026 - Kanerika, 1月 9, 2026にアクセス、 <https://kanerika.com/blogs/chatgpt-vs-gemini-vs-claude/>
16. Gemini 2.0 model updates: 2.0 Flash, Flash-Lite, Pro Experimental - Google Blog, 1月 9, 2026にアクセス、 <https://blog.google/technology/google-deepmind/gemini-model-updates-february-2025/>
17. Long context | Gemini API - Google AI for Developers, 1月 9, 2026にアクセス、 <https://ai.google.dev/gemini-api/docs/long-context>
18. Long context | Generative AI on Vertex AI - Google Cloud Documentation, 1月 9, 2026にアクセス、 <https://docs.cloud.google.com/vertex-ai/generative-ai/docs/long-context>
19. What is The Best AI Model in 2025/2026? | AI Hub, 1月 9, 2026にアクセス、 <https://overchat.ai/ai-hub/the-best-ai-model>
20. Part 4 | ChatGPT Canvas or Claude Artifacts: How to Create SQL Database & Python Code from Simple Images - AI Fire, 1月 9, 2026にアクセス、 <https://www.aifire.co/p/detailed-comparison-for-interactive-tools-canvas-or-artifacts>
21. AI Coding Tools Showdown: Cursor vs Bolt vs Windsurf | UI Bakery Blog, 1月 9, 2026にアクセス、 <https://uibakery.io/blog/cursor-vs-bolt-vs-windsurf>
22. Windsurf Editor, 1月 9, 2026にアクセス、 <https://windsurf.com/editor>
23. An Update to Our Free Plan - Windsurf, 1月 9, 2026にアクセス、 <https://windsurf.com/blog/update-to-free-plan>
24. Kidjig/bolt.diy: Prompt, run, edit, and deploy full-stack web applications using any LLM you want! - GitHub, 1月 9, 2026にアクセス、 <https://github.com/Kidjig/bolt.diy>
25. stackblitz-labs/bolt.diy: Prompt, run, edit, and deploy full-stack web applications using any LLM you want! - GitHub, 1月 9, 2026にアクセス、 <https://github.com/stackblitz-labs/bolt.diy>
26. DeepSeek-R1 671B: Complete Hardware Requirements - DEV Community, 1月 9, 2026にアクセス、 <https://dev.to/askyt/deepseek-r1-671b-complete-hardware-requirements-optimal-deployment-setup-2e48>
27. Models: 'deepseek/deepseek-r1' - OpenRouter, 1月 9, 2026にアクセス、 <https://openrouter.ai/deepseek%2Fdeepseek-r1>
28. Run DeepSeek R1 locally on your device (Beginner-Friendly Guide) - Jan.ai, 1月 9, 2026にアクセス、 <https://www.jan.ai/post/deepseek-r1-locally>
29. Gemini Developer API pricing, 1月 9, 2026にアクセス、 <https://ai.google.dev/gemini-api/docs/pricing>
30. Is there a free tier and what are its limits? - Groq Community, 1月 9, 2026にアクセス、 <https://community.groq.com/t/is-there-a-free-tier-and-what-are-its-limits/790>
31. Free AI Models on OpenRouter, 1月 9, 2026にアクセス、 <https://openrouter.ai/collections/free-models>